University of Exeter
Department of Mathematics

A comparison of polynomial chaos and Gaussian process emulation for uncertainty quantification in computer experiments

Nathan Edward Owen

May 2017

Supervised by
Peter Challenor
Prathyush Menon

Submitted by Nathan Edward Owen, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Mathematics , May 2017.

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Abstract

Computer simulation of real world phenomena is now ubiquitous in science, because experimentation in the field can be expensive, time-consuming, or impossible in practice. Examples include climate science, where future climate is examined under global warming scenarios, and cosmology, where the evolution of galaxies is studied from the beginning of the universe to present day. Combining complex mathematical models and numerical procedures to solve them in a computer program, these *simulators* are computationally expensive, in that they can take months to complete a single run.

The practice of using a simulator to understand reality raises some interesting scientific questions, and there are many sources of uncertainty to consider. For example, the discrepancy between the simulator and the real world process. The field of uncertainty quantification is concerned with the characterisation and reduction of all uncertainties present in computational and real world problems. A key bottleneck in any uncertainty quantification analysis is the cost of evaluating the simulator. The solution is to replace the expensive simulator with a *surrogate model*, which is computationally faster to run, and can be used in subsequent analyses.

Polynomial chaos and Gaussian process emulation are surrogate models developed independently in the engineering and statistics communities respectively over the last 25 years. Despite tackling similar problems in the field, there has been little interaction and collaboration between the two communities. This thesis provides a critical comparison of the two methods for a range of criteria and examples, from simple test functions to simulators used in industry. Particular focus is on the approximation accuracy of the surrogates under changes in the size and type of the experimental design. It is concluded that one method does not unanimously outperform the other, but advantages can be gained in some cases, such that the preferred method depends on the modelling goals of the practitioner. This is the first direct comparison of polynomial chaos and Gaussian process emulation in the literature.

This thesis also proposes a novel methodology called probabilistic polynomial chaos, which is a hybrid of polynomial chaos and Gaussian process emulation. The approach draws inspiration from an emerging field in scientific computation known as

probabilistic numerics, which treats classical numerical methods as statistical inference problems. In particular, a probabilistic integration technique called Bayesian quadrature, which employs Gaussian process emulators, is applied to a traditional form of polynomial chaos. The result is a probabilistic version of polynomial chaos, providing uncertainty information where the simulator has not yet been run.

Thank you to my supervisors, Peter Challenor and Prathyush Menon, for giving me the opportunity to do the PhD and supporting me every step of the way.

Thank you to my family: Mum, Dad, and Jack. You were always there for me, whether it be on the phone for a chat or driving down the A303 to come and see me. Mum, I promise to come home more often from now on.

Thank you to my housemates, Ed and Gabby, for making the last few years really enjoyable at home. Thanks also to my two furry housemates, Pippa and Columbus, for providing so much entertainment — your early morning scratching and meowing was just the motivation I needed to get up and write the thesis!

Thank you to all my friends and colleagues over the last few years, who have made my time in Exeter so enjoyable. Special mention goes to James, Laura and Nina who have been there from start to finish.

Finally, thank you to Georgina for all your love and support. Thanks for your patience, and pretending to understand, on the several occasions I tried to explain my work to you!

Contents

List of tables **10**

List of figures **11**

Publications **19**

1. Introduction **21**
1.1. Aims . . . 23
1.2. Structure of this thesis . . . 23

2. Background **25**
2.1. Simulators and computer experiments . . . 25
2.2. Uncertainty quantification . . . 27
2.2.1. Objectives in uncertainty quantification . . . 28
2.2.2. Traditional methods . . . 30
2.3. Surrogate methods . . . 31
2.3.1. Polynomial chaos . . . 32
2.3.2. Gaussian process emulation . . . 40
2.3.3. Comparisons and hybrid approaches . . . 46
2.4. Summary . . . 48

3. Surrogate modelling **50**
3.1. Introduction and notation . . . 50
3.2. Polynomial chaos . . . 53
3.2.1. Choosing the polynomial basis . . . 54
Orthogonal polynomials . . . 54
Polynomial chaos expansion . . . 59
Truncating the polynomial chaos expansion . . . 60
3.2.2. Finding the coefficients . . . 63
Regression . . . 63
Non-intrusive spectral projection . . . 64
3.2.3. Polynomial chaos surrogate . . . 67
Statistical moments . . . 67
3.2.4. Examples . . . 68

3.3. Gaussian process emulation . . . 72
3.3.1. Gaussian process prior . . . 73
Mean function . . . 73
Covariance function . . . 74
Examples . . . 76
3.3.2. Deriving the posterior . . . 76
3.3.3. Gaussian process surrogate . . . 81
3.3.4. Examples . . . 82
3.4. Discussion . . . 86

4. Comparison of surrogate methods **93**
4.1. Surrogate model validation . . . 93
4.1.1. Polynomial chaos . . . 95
4.1.2. Gaussian process emulation . . . 96
4.2. Methods for comparing surrogates . . . 98
4.2.1. Oakley and O'Hagan (2002) method . . . 101
4.3. Experiments . . . 102
4.3.1. adJULES simulator . . . 102
4.3.2. VEGACONTROL simulator . . . 103
4.3.3. Experimental set-up . . . 105
4.4. Results . . . 107
4.4.1. adJULES experiment . . . 107
4.4.2. VEGACONTROL experiment . . . 113
4.4.3. Discussion . . . 117
4.5. General advantages and disadvantages . . . 118
4.6. Conclusion . . . 122

5. Probabilistic numerics and polynomial chaos **125**
5.1. Probabilistic numerics . . . 125
5.1.1. Introduction . . . 125
5.1.2. Bayesian quadrature . . . 128
5.1.3. Implementation . . . 131
Analytical solutions to $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ . . . 131
Experimental design . . . 134
5.1.4. Examples . . . 135
5.2. Probabilistic polynomial chaos . . . 136
5.2.1. Application of Bayesian quadrature . . . 139
5.2.2. Implementation . . . 142
Properties of $\psi(\mathbf{x})\eta(\mathbf{x})$ . . . 142
Analytical solutions to $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ . . . 143
Recovering traditional polynomial chaos . . . 147

Experimental design 150
5.2.3. Probabilistic polynomial chaos surrogate 151
Statistical moments 152
5.2.4. Examples 153
5.3. Discussion 158
5.4. Conclusion 162

6. Conclusion **164**
6.1. Summary 164
6.2. Directions for future development 167
6.3. Conclusion 169

Appendices **171**

A. Additional figures for the probabilistic polynomial chaos examples **172**

B. Comments from the viva voce examination with response from the author **182**
B.1. Chapter 1 182
B.2. Chapter 2 182
B.3. Chapter 3 185
B.4. Chapter 4 187
B.5. Chapter 5 189
B.6. Chapter 6 192

Bibliography **194**

List of Tables

2.1. Correspondence of orthogonal polynomial families from the Askey scheme to random variable distributions in generalised polynomial chaos ($N \geq 0$ is a finite integer). Adapted from Xiu and Karniadakis (2003). 35

4.1. Summary of the designs used in the adJULES and VEGACONTROL experiments. The design size n increases to allow polynomial chaos expansions with larger truncation orders p. A description of the design classes is given in Section 4.3.3. Note that a fourth class 3 design was not implemented for the VEGACONTROL experiment due to computational restrictions. Sobol sequence designs of size 625 and 1024 were also used for the adJULES and VEGACONTROL experiments respectively for comparison to the largest tensor grid designs. 105

5.1. The construction of the modified Hermite polynomials, $H^*_\alpha(x, y)$, $\alpha = 0, \ldots, 5$, using the standard Hermite polynomials, $H_\alpha(x)$, and the polynomial $\sum_{i=0}^{|H_\alpha(x)|-1} y^i$, where $|H_\alpha(x)|$ denotes the number of terms in $H_\alpha(x)$. 147

List of Figures

3.1. The Legendre polynomials, $P_k(x)$, for $k = 0, \ldots, 5$. 57

3.2. The Hermite polynomials, $H_k(x)$, for $k = 0, \ldots, 5$. 57

3.3. The logarithm of the number of terms in the truncated polynomial chaos expansion, N, as a function of the input dimension d and the polynomial order p, for total order (left panel) and tensor product (right panel) truncation schemes. 62

3.4. Polynomial chaos surrogates (blue) for the one-dimensional example $y = \eta(x) = 5 + 5x + \sin(5 + 5x)$ (black). The rows correspond to truncation orders $p = 1, 2, 3, 4$, which are built using design sizes $n = 2, 3, 4, 5$, and the columns refer to the regression and non-intrusive spectral projection (NISP) techniques for estimating the coefficients. Regression and NISP polynomial chaos surrogates are built on Sobol sequence and Gauss-Legendre quadrature rule designs respectively. The experimental design is shown as black dots in each case. 69

3.5. Polynomial chaos surrogates (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image). The rows correspond to truncation orders $p = 1, 2, 3, 4$, which are built using design sizes $n = 4, 9, 16, 25$, and the columns refer to the regression and non-intrusive spectral projection (NISP) techniques for estimating the coefficients. Regression and NISP polynomial chaos surrogates are built on Sobol sequence and tensor grid designs respectively. The experimental design is shown as black dots in each case. 71

3.6. Five samples from one-dimensional Gaussian process priors (black lines) with squared exponential correlation functions, using different settings for the variance σ^2 and correlation length δ hyperparameters. The mean function is a constant $\beta_1 = 0$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sigma$ (grey shading), in each case. 77

3.7. Five samples from one-dimensional Gaussian process priors (black lines) with Matérn correlation functions (shape parameter $\nu = 5/2$), using different settings for the variance σ^2 and correlation length δ hyperparameters. The mean function is a constant $\beta_1 = 0$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sigma$ (grey shading), in each case. 78

3.8. Gaussian process posterior mean functions (blue) for the one-dimensional example $y = \eta(x) = 5+5x+\sin(5+5x)$ (black), using Sobol sequence designs. For all cases, the prior mean function is a constant $\beta_1 = 0$. Posterior uncertainty is represented as two posterior standard deviations around the posterior mean, $M^{**}(x) \pm 2\sqrt{V^{**}(x,x)}$, and plotted across the input domain $x \in [-1, 1]$ (grey shading). The rows correspond to design sizes of $n = 2, 3, 4, 5$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 84

3.9. Gaussian process posterior mean functions (blue) for the one-dimensional example $y = \eta(x) = 5 + 5x + \sin(5 + 5x)$ (black), using Gauss-Legendre quadrature rule designs. Posterior uncertainty is represented as two posterior standard deviations around the posterior mean, $M^{**}(x) \pm 2\sqrt{V^{**}(x,x)}$, and plotted across the input domain $x \in [-1, 1]$ (grey shading). The rows correspond design sizes of $n = 2, 3, 4, 5$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 85

3.10. Gaussian process posterior mean functions (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image), using Sobol sequence designs. For all cases, the prior mean function is a constant $\beta_1 = 0$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 87

3.11. Gaussian process posterior uncertainty (coloured image) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (not shown), using Sobol sequence designs. Posterior uncertainty is represented as two posterior standard deviations (2 SD), $2\sqrt{V^{**}(\mathbf{x},\mathbf{x})}$, and plotted across the input domain $\mathbf{x} = (x_1, x_2) \in [-1, 1]^2$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 88

3.12. Gaussian process posterior mean functions (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image), using tensor grid designs of Gauss-Legendre quadrature rules. For all cases, the prior mean function is a constant $\beta_1 = 0$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 89

3.13. Gaussian process posterior uncertainty (coloured image) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (not shown), using tensor grid designs of Gauss-Legendre quadrature rules. Posterior uncertainty is represented as two posterior standard deviations (2 SD), $2\sqrt{V^{**}(\mathbf{x}, \mathbf{x})}$, and plotted across the input domain $\mathbf{x} = (x_1, x_2) \in [-1, 1]^2$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case. 90

4.1. Visualisation of the adJULES simulator output (cost function) as a function of pairwise combinations of each of the four inputs. The smoothing of the simulator output is performed using the surrogate with the lowest root mean square error in the adJULES experiments (see Figure 4.3), which is the the polynomial chaos expansion built on the largest class 3 design (see Table 4.1). 103

4.2. Visualisation of the VEGACONTROL simulator output ($\max(Q_\alpha)$) as a function of pairwise combinations of each of the five inputs. The smoothing of the simulator output is performed using the surrogate with the lowest root mean square error in the VEGACONTROL experiments (see Figure 4.7), which is the Gaussian process emulator with Matérn correlation function built on the largest class 2 design (see Table 4.1). 104

4.3. Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the adJULES simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines). 109

4.4. Exceedance probabilities $Pr(Y > \mu_Y + 2\sigma_Y)$ and $Pr(Y > \mu_Y + 3\sigma_Y)$ as a function of design size n and design class for the adJULES simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines). 110

4.5. Probability density function (PDF) validation metrics for the adJULES simulator. Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates. . 111

4.6. Difference in root mean square error (RMSE), mean and standard deviation validation metrics when surrogates are built on large Sobol sequence designs instead of tensor grids. Sobol sequences of size 625 and 1024 were used in the adJULES (top panels) and VEGACONTROL (bottom panels) experiments respectively, to match the largest tensor grid designs. Validation metrics from the tensor grid designs are shown in black, whereas metrics from the Sobol sequence designs are shown in colour. 112

4.7. Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the VEGACONTROL simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines). 114

4.8. Exceedance probabilities $Pr(Y > \mu_Y+2\sigma_Y)$ and $Pr(Y > \mu_Y+3\sigma_Y)$ as a function of design size n and design class for the VEGACONTROL simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines). 115

4.9. Probability density function (PDF) validation metrics for the VEGACONTROL simulator. Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates. 116

4.10. Plot of the two-dimensional toy simulator in Equation (4.14) for $x_1, x_2 \in [-1, 1]$. 120

4.11. Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the two-dimensional toy simulator in Equation (4.14). The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines). 121

4.12. Exceedance probabilities $Pr(Y > \mu_Y + 2\sigma_Y)$ and $Pr(Y > \mu_Y + 3\sigma_Y)$ as a function of design size n and design class for the two-dimensional toy simulator in Equation (4.14). The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines). 122

4.13. Probability density function (PDF) validation metrics for the two-dimensional toy simulator in Equation (4.14). Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates. 123

5.1. Bayesian quadrature for the one-dimensional example in Equation (5.27) using Monte Carlo samples of size $n = 5, 10, 15$ (columns). Top row: three draws from the GP prior for $g(x)$ (black dashed lines), posterior GP mean (blue line) with 95% credible intervals (grey shading) for $g(x)$ given the samples (black dots), standard Gaussian distribution density for reference (red line, not to scale). Bottom row: prior (black dashed line) and posterior (blue shading) distributions for $\mathcal{I}$ with true value (black vertical line). 136

5.2. Bayesian quadrature for the one-dimensional example in Equation (5.27) using Quasi Monte Carlo samples of size $n = 5, 10, 15$ (columns). Top row: three draws from the GP prior for $g(x)$ (black dashed lines), posterior GP mean (blue line) with 95% credible intervals (grey shading) for $g(x)$ given the samples (black dots), standard Gaussian distribution density for reference (red line, not to scale). Bottom row: prior (black dashed line) and posterior (blue shading) distributions for $\mathcal{I}$ with true value (black vertical line). 137

5.3. Five draws from one-dimensional Gaussian process priors (black) modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$. The mean function is a constant $\beta_1 = 0$ and the covariance function is squared exponential with variance $\sigma^2 = 1$ and correlation length $\delta = 1$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sqrt{\text{var}\left[H_\alpha(x)\eta(x)\right]}$ (grey shading), in each case. 144

5.4. Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a QMC sample from a Sobol sequence of size $n = 10$ (black dots). 155

5.5. Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 10$ is used. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line. 156

5.6. A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 10$ is used. 157

5.7. Probabilistic polynomial chaos surrogates for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$ for the example in Section 5.2.4. The rows correspond to the two design strategies: Quasi Monte Carlo samples from a Sobol sequence (top row) and Gauss-Hermite quadrature rules combined with Quasi Monte Carlo samples (bottom row). The columns correspond to design sizes of $n = 10, 15$. In each case, the posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The traditional polynomial chaos surrogate, built using a Gauss-Hermite quadrature rule of size $n = 6$ is also shown (green), as well as the true function (black). The experimental design points are shown as black dots. 159

A.1. Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a QMC sample from a Sobol sequence of size $n = 15$ (black dots). 173

A.2. Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 15$ is used. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line. 174

A.3. A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 15$ is used. 175

A.4. Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4, to give a design size $n = 10$ (black dots). 176

A.5. Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4 is used, to give a design size $n = 10$. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line. 177

A.6. A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4 is used, to give a design size $n = 10$. 178

A.7. Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with a posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9, to give a design size $n = 15$ (black dots). 179

A.8. Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9 is used, to give a design size $n = 15$. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line. 180

A.9. A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9 is used, to give a design size $n = 15$. 181

Publications

Material from Chapter 4 has been published in:

Owen, N. E., P. Challenor, P. P. Menon, and S. Bennani, 2017: Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators. *SIAM/ASA Journal on Uncertainty Quantification*, **5**, 403–435.

1. Introduction

Physical systems — such as weather, climate, cosmology and construction — are routinely studied using computer models, known as simulators. This is because direct experimentation in the real world may be expensive, impractical, dangerous, time-consuming or even impossible. Simulators describe a physical system of interest using a set of equations, usually comprising a complex system of ordinary or partial differential equations, which represent the current state of the art of scientific knowledge about the real world process. These equations are often in terms in a large number of inputs and outputs, which may represent real world quantities or parameterisations of them. To run the simulator at a particular choice of the inputs, the equations must be solved numerically using a computational procedure. The evaluation of the corresponding outputs may take a substantial amount of time — hours, days, weeks, even when using a supercomputer — because of the inherent complexity of the equations and the numerical methods needed to solve them. As a consequence, a practitioner will only be able to complete a finite number of simulator runs, due to restrictions on time or computational resources. The specific input settings at which the simulator is run is known as a computer experiment (Sacks et al., 1989), and must be chosen carefully in order to best learn about the physical system of interest.

The field of uncertainty quantification (UQ) is concerned with the characterisation and reduction of all uncertainties present in computational and real world problems (Sullivan, 2015). For UQ in computer experiments, the practice of using a simulator to learn about the real world introduces many sources of uncertainty (Kennedy and O'Hagan, 2001). For example, the uncertainty in specifying the correct settings for the simulator inputs (since they may take a range of plausible values) is known as parameter uncertainty. Also, the output of the simulator will never match reality, even if there is no parameter uncertainty (due to an incomplete specification of the equations, an error in their numerical solution, or a mismatch in temporal or spatial resolutions), and this is called model discrepancy. Furthermore, if measurements of the real world process are to be used in any way, it is likely that they have been made with observational error. The uncertainty arising from these sources, amongst many others, must be quantified properly if any inference from the simulator about the real world is to be viable.

There are several objectives in UQ for computer experiments, which use the simulator in different ways to learn about the physical system of interest. Essentially there are two coupled problems: the forward problem, where uncertainty in the inputs is propagated through the simulator to the outputs to learn about the input-output relationship of the physical process; and the inverse problem, where observations of the physical process are used to constrain the simulator to match reality, under the presence of uncertainty. Within these two cases, there are a number of specific objectives. Given a computer experiment, the prediction objective is concerned with estimating the value of the simulator output at a new input setting (Sacks et al., 1989). An uncertainty analysis studies how uncertainty in the inputs propagates through the simulator to the outputs (Oakley and O'Hagan, 2002; Xiu and Karniadakis, 2003). Related to this is a sensitivity analysis, which aims to identify the most influential inputs in driving changes in the simulator outputs (Saltelli et al., 2009). Calibration (Kennedy and O'Hagan, 2001) and history matching (Craig et al., 1996) are inverse problems which use observational data to tune the simulator; that is, find input settings that produce simulator output consistent with reality.

A bottleneck in carrying out any UQ objective is the computational and time expense required to repeatedly run the simulator. Traditional methods for solving these problems, such as Monte Carlo simulation (Caflisch, 1998), are ruled out because the number of available runs of the simulator is too small to obtain good accuracy. A more sophisticated solution from the UQ literature is to build a surrogate model, which acts as an approximation to the simulator. Constructed using information from the computer experiment, the surrogate aims to imitate the behaviour of the simulator as closely as possible while being computationally faster to evaluate. The surrogate can then be used in place of the simulator in subsequent analyses for a reduction in the computational burden. The process of replacing the simulator with a surrogate introduces an additional source of uncertainty, but this can be quantified appropriately in the spirit of UQ.

Many types of surrogate models exist in the UQ literature. Two of the most popular surrogate approaches in UQ for computer experiments, polynomial chaos (Ghanem and Spanos, 1991b; Xiu and Karniadakis, 2003) and Gaussian process emulation (Santner et al., 2003; Rasmussen and Williams, 2006), will be the main focus of this thesis. Polynomial chaos originates in the engineering and applied mathematics communities, and represents the simulator output using a series expansion of orthogonal polynomials in terms of the simulator inputs (Ghanem and Spanos, 1991b). On the other hand, Gaussian process emulation is a statistical approach which treats the simulator as an unknown function that can be modelled as a realisation of a stochastic process (Sacks et al., 1989). Both polynomial chaos and Gaussian process emulation have been developed and successfully applied for UQ in computer

experiments over roughly the same time period, that is, the last 25 years.

Despite providing efficient solutions to essentially the same problems in UQ, there has been a surprising lack of interaction between the polynomial chaos and Gaussian process emulation communities. In particular, very little research has been carried out to critically compare the methods, meaning that it is unclear to the UQ community as a whole which surrogate technique may perform better in different scenarios. It may also be possible to combine polynomial chaos and Gaussian process emulation to form a hybrid surrogate approach, which might draw upon the individual advantages of each method.

1.1. Aims

The main objective of this thesis is to critically compare polynomial chaos and Gaussian process emulation, in order to provide useful information for practitioners in UQ as to which performs better in different modelling scenarios, and their respective advantages and disadvantages in general. The possibility of developing a hybrid approach of polynomial chaos and Gaussian process emulation will also be examined.

More specifically, the following questions will be addressed:

- How accurate are the surrogate methods in approximating the simulator?
- How does this accuracy change with variations in the form of the computer experiment (for example the size and type of the experimental design)?
- Is it the case that one surrogate unanimously outperforms the other, or does the preferred method depend on the modelling scenario?
- How do the surrogates compare in terms of other important criteria, for example: the computational costs of building the surrogate; the flexibility and practicality of the surrogate; and ease of implementation?
- Is it possible to combine the surrogates to form a hybrid approach?

1.2. Structure of this thesis

The structure of the thesis is as follows. Chapter 2 is a background and literature review of computer experiments, uncertainty quantification and surrogate modelling. Particular focus is on the developments of polynomial chaos and Gaussian process emulation over the last 25 years. Discussion of existing comparisons and hybrid

approaches of the two methods is also given, showing that these are rare in the literature.

Chapter 3 introduces the mathematical notation for surrogate modelling and outlines a number of properties a good surrogate should possess. Following this, a detailed mathematical description of polynomial chaos and Gaussian process emulation is presented. The two surrogate approaches are then applied to simple one-dimensional and two-dimensional test functions for demonstrative purposes, as well as motivating a comparison.

Chapter 4 concerns the validation of surrogate models, that is, how the quality of the surrogate model in approximating the simulator is assessed in the polynomial chaos and Gaussian process emulation communities. Since this is done differently in each community, an unbiased set of validation metrics for comparing the surrogate approaches to one another is proposed. These metrics are used to compare the accuracy of the surrogates in approximating two simulators used in industry. Particular interest is in comparing the accuracy under changes in the size and type of the design of the computer experiment. The chapter is concluded with a discussion of the comparative advantages and disadvantages of the two methods for a range of criteria.

Chapter 5 presents a novel hybrid surrogate model, combining polynomial chaos and Gaussian process emulation, called probabilistic polynomial chaos. The proposed approach rectifies a disadvantage of traditional polynomial chaos highlighted in the preceding chapters, in that it is a fully probabilistic surrogate for the simulator, providing uncertainty about its predictions. The methodology draws upon techniques from an emerging field in scientific computation, known as probabilistic numerics, which treats classical numerical methods as statistical inference problems. The probabilistic polynomial chaos surrogate is tested for a simple one-dimensional example.

Finally, Chapter 6 concludes the thesis with a summary of the key findings from the preceding chapters, as well as giving directions for future development of the work.

2. Background

This chapter is an introduction and literature review of the key concepts and techniques, and is structured as follows. Simulators and computer experiments are introduced in Section 2.1, followed by the related field of uncertainty quantification in Section 2.2. Various problem objectives exist within uncertainty quantification, and these are presented in Section 2.2.1, along with traditional methods for solving them in Section 2.2.2. These traditional methods have a number of disadvantages, which modern techniques based on surrogate modelling attempt to rectify. Surrogate models are reviewed in Section 2.3 and two of the most popular techniques, known as polynomial chaos and Gaussian process emulation, are described in detail in Sections 2.3.1 and 2.3.2 respectively. A small number of comparisons and combinations of the two methods exist in the literature, and these are briefly reviewed in Section 2.3.3. Finally, the chapter is concluded with a summary in Section 2.4, where the need for a critical comparison of polynomial chaos and Gaussian process emulation is highlighted.

2.1. Simulators and computer experiments

Computer models, or simulators, are now used in virtually all areas of science. They provide a means for studying a physical system of interest without having to experiment in the field. Examples include: climate science (Gordon et al., 2000; Bellouin et al., 2011), where future states of the climate system are examined under global warming emission scenarios; cosmology (Bower et al., 2006), where the creation and evolution of galaxies from the beginning of the universe to present day are studied using galaxy formation models; social science (Sun, 2006), where agent-based models simulate the actions and interactions of individuals and their contribution to the system as a whole; and engineering (Kirkpatrick, 2000), where structures for real world problems (automobile, aerospace and construction) are designed and tested before they are built. In many cases, direct experimentation in the real world may be expensive, impractical, dangerous, time-consuming, or even impossible. Simulators can act as a viable, safer and cheaper alternative.

Simulators are generally made up of two components. Firstly, a mathematical model

usually exists which describes the physical process completely or partially, according the current state of theoretical knowledge. This mathematical model usually comprises a set of complex ordinary or partial differential equations which govern the behaviour of the system in a spatial or temporal domain of interest. These equations are typically in terms of a large number of inputs and outputs, which may represent real world quantities, or parameterisations of them. Secondly, because the mathematical model often cannot be solved analytically it must be implemented in a computational model. The computer code to do this may run to millions of lines long, and commonly will use numerical methods (finite element solvers) to integrate the differential equations over a spatial and/or time domain. Due to complexity of both the mathematical model and the computational tools needed to solve it, the simulator may take a substantial amount of time to complete a single run at a given input configuration, even when using a supercomputer. For example, a global climate model may take several months to complete a single run (Rougier et al., 2009b).

Simulators are often treated as "black-box" models — a system considered just in terms of its inputs and outputs, without any knowledge of its inner workings. This can be the case even when the mathematical model component is known in full, because the inherent complexity means that its solution is not really known until the simulator is run. In this case, it is common to write the simulator simply as a functional relationship, $\mathbf{y} = \eta(\mathbf{x})$. That is, a black-box function $\eta(\cdot)$ linking a high dimensional set of inputs $\mathbf{x}$ to a high dimensional set of outputs $\mathbf{y}$. The simulator can be run at a chosen input setting, say $\mathbf{x}^{(i)}$, producing an output $\mathbf{y}^{(i)}$. Many simulators are deterministic in the sense that an identical output will be produced when the simulator is run at the same input settings. The alternative to this is a stochastic simulator (Mortensen and Haggerty, 1988), where the output includes some randomness due to a stochastic component within the system.

In practice, one will have a fixed budget for running a simulator, arising from a mixture of computational, time or expense restrictions. Because of this, a practitioner will only have access to a fixed finite number of simulator evaluations, which must be chosen wisely. In general terms, the simulator should be run at a set of input locations, $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$ (known as the experimental design), which allow us to best learn about the physical system of interest. After running the simulator at each input setting in $\mathcal{D}$, the corresponding output is $\mathbf{y} = \left(\mathbf{y}^{(1)} = \eta(\mathbf{x}^{(1)}), \ldots, \mathbf{y}^{(n)} = \eta(\mathbf{x}^{(n)})\right)$. The data arising from the choice of input locations in $\mathcal{D}$ and their corresponding simulator output values in $\mathbf{y}$ is known as a computer experiment. This is as opposed to a physical experiment, where data would be collected in the field or laboratory. The design and the analysis of computer experiments is an established and constantly growing scientific field (Santner

et al., 2003).

2.2. Uncertainty quantification

The practice of using a simulator to study reality in a computer experiment poses some interesting scientific questions. For example, how can one be sure that the simulator accurately portrays the real world phenomena? There are many uncertainties present in this framework, arising from unknowns in the construction of the simulator, the numerical solution of the mathematical component, as well as the incorporation of real world data. The field of uncertainty quantification (UQ) aims to address these issues. UQ is the science of the characterisation and reduction of uncertainties present in computational and real world problems (Smith, 2013). Sources of uncertainty in computer experiments may be categorised as (Kennedy and O'Hagan, 2001):

- **Parameter uncertainty**: the input settings of the simulator must be specified if it is to be run. While some inputs are fixed because they represent known processes (for example, acceleration due to gravity $g = 9.8\text{ms}^{-2}$), many input settings will be unknown and may take a range of plausible values. Parameter uncertainty refers to the uncertainty concerning the correct settings of the inputs, and may be represented as an upper and lower bound, a mean and a variance, or a full probability distribution. Parameter uncertainty also includes the specification of initial or boundary conditions for the simulator.

- **Model discrepancy**: even if there is no parameter uncertainty, so that the true values of the inputs are known completely, the simulator output will not match reality. This may be due to an incomplete specification of the mathematical model, numerical errors made in the computational code, or inferior spatial or temporal resolution. The difference between the true process and the simulator is known as model discrepancy or model inadequacy.

- **Residual variability**: the simulator is in effect trying to predict the value of a physical system given a set of inputs. In reality the true process may vary every time the experiment is repeated at those inputs. This variation may be due to the fact that the process is actually stochastic, or because the set of inputs to the simulator are insufficiently detailed (and the variability could be reduced with the inclusion of more inputs). Residual variability deals with the uncertainty in the true process arising from these two issues.

- **Parametric variability**: linked to residual variability, but the case where there is not enough information to specify the inputs, so that the simulator

output depends on some uncontrolled and unspecified conditions. This induces an extra source of uncertainty into the predicted simulator output which is known as parametric variability.

- **Observational error**: measurements of the physical process may be used to validate the simulator or constrain it in some way. It is likely that these are made with an amount of error, which is an extra source of uncertainty known as observational error.

- **Code uncertainty**: in principle, the output of the simulator at a given set of inputs is known because it is a known function of the inputs. However, due to the complexity of the simulator and the time it takes to run, simulator output is not really known until the simulator is actually evaluated. Since it is not practical to run the simulator at every available input configuration, uncertainty in the output for untried input settings must be accounted for and is known as code uncertainty.

Researchers in UQ advocate that any design and analysis of computer experiments should account for the appropriate sources of uncertainty listed above, in order for it to be a complete and viable piece of work. Representing this uncertainty accuracy is another issue, and is commonly done using expert elicitation, introducing yet another source of uncertainty. The inclusion of uncertainty will naturally lead to a more conservative analysis, but will reduce the chance of being overconfident about the wrong conclusion. This is particularly important when simulators are part of a decision making process, for example policy making for government.

2.2.1. Objectives in uncertainty quantification

There are essentially two coupled problems in UQ for computer experiments. The first is known as the forward problem, where uncertainty arising from the input specification is to be propagated through the simulator to the outputs. The second is the inverse problem, where observational data of the physical process is to be used to constrain the simulator in some way to match reality, under the presence of the uncertainties outlined in Section 2.2. Within these two problems, however, there are various objectives in any UQ analysis, which will now be briefly described.

- **Prediction**: given a computer experiment comprising a experimental design $\mathcal{D}$ and corresponding output $\mathbf{y}$, the prediction objective is concerned with estimating the value of the simulator output at a new input setting $\mathbf{x}^*$ (Sacks et al., 1989).

- **Uncertainty analysis**: this objective, also known as uncertainty propaga-

tion, investigates how uncertainty in the inputs (mainly parameter uncertainty) propagates through the simulator to the output. Statistical summaries of the output are desired and must be estimated using evaluations of the simulator. This may include its mean or variance, the probability of exceeding a threshold, or its cumulative distribution function.

- **Sensitivity analysis**: related to an uncertainty analysis, this objective aims to identify the most influential inputs in driving changes in the simulator outputs, under the presence of input uncertainty. This allows for uncertainty in the output to be reduced with more resources directed at the most sensitive inputs. It also may be used as a screening experiment, to identify the active inputs for a reduction of model complexity. There is a large literature on sensitivity analysis (Saltelli et al., 2009), and a number of sensitivity indices can be estimated to this end. These include local methods which study the derivative of the output with respect to each of the inputs, and global methods which decompose the output variance into contributions from each of the inputs.

- **Calibration and history matching**: the inverse problem, where observational data is used to tune the simulator to match reality. Calibration involves the process of finding the best input settings that result in simulator outputs consistent with observational data (Kennedy and O'Hagan, 2001). An alternative strategy, known as history matching (Craig et al., 1996), takes a different approach by ruling out regions of input space that lead to simulator outputs inconsistent with observational data. A major source of uncertainty here is the model discrepancy.

- **Optimisation**: this objective is concerned with finding the simulator input settings which lead to outputs that satisfy some criterion. A common example is identifying the input setting which maximises or minimises the simulator output in some range. In this sense, optimisation is similar to calibration but without the use of observational data.

- **Design**: in all of the above objectives, a key challenge involves the choice of the experimental design $\mathcal{D}$. Ideally, the input locations should be chosen such that the maximum amount of information is contained in the corresponding simulator output to perform a uncertainty or sensitivity analysis, calibration or optimisation. However, this is not a trivial task and selecting a experimental design that is "optimal" in some way can be considered an objective in itself.

2.2.2. Traditional methods

Traditional methods for solving the UQ objectives in Section 2.2.1 mainly revolve around Monte Carlo (MC) simulation (Caflisch, 1998). In general, this method works by obtaining a large sample from a distribution and taking a statistical average of a function of that sample. In the context of computer experiments, suppose the simulator inputs, $\mathbf{x}$, are uncertain and can be represented by the random vector, $\mathbf{X}$, with probability distribution $p(\mathbf{X})$. Suppose also that it is possible to obtain samples from the distribution $p(\mathbf{X})$. By construction the outputs of the simulator, $\mathbf{y}$, are a random vector, $\mathbf{Y} = \eta(\mathbf{X})$, with unknown distribution $p(\mathbf{Y})$. MC simulation can be used to tackle the UQ objectives outlined in Section 2.2. For example, a MC simulation approach for the uncertainty analysis of the simulator would proceed as follows:

1. Draw a random sample $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \sim p(\mathbf{X})$.

2. Evaluate the simulator for each to produce $\mathbf{y}^{(1)} = \eta(\mathbf{x}^{(1)}), \ldots, \mathbf{y}^{(n)} = \eta(\mathbf{x}^{(n)}) \sim p(\mathbf{Y})$.

3. Estimate the expectation of any statistical summary, $f(\mathbf{Y})$, of the output as the average:
$$\mathbb{E}\left[f(\mathbf{Y})\right] \approx \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{y}^{(i)}). \tag{2.1}$$

For example, the expectation of $\mathbf{Y}$ is approximated as the sample mean $\bar{\mathbf{y}}$:

$$\mathbb{E}\left[\mathbf{Y}\right] \approx \bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}^{(i)}, \tag{2.2}$$

and the covariance of $\mathbf{Y}$ is estimated using the sample covariance matrix:

$$\operatorname{cov}\left[\mathbf{Y}\right] \approx \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{y}^{(i)} - \mathbf{y})(\mathbf{y}^{(i)} - \mathbf{y})^{\top}. \tag{2.3}$$

MC solutions to the other UQ objectives in Section 2.2.1 work along the same principle. The main advantages of MC methods are their simplicity and versatility. Furthermore, bootstrap techniques (Efron and Tibshirani, 1994) can be easily applied to estimate uncertainty information, for example confidence intervals, for MC estimates. However, MC methods are known to have a slow convergence rate of $\mathcal{O}(1/\sqrt{n})$ (Caflisch, 1998), meaning that a large sample size n is required to ensure good accuracy (though this is independent of the dimension of the inputs). This rules out MC methods for expensive simulators, where obtaining a large number of simulator runs is simply not possible due to time and computational restraints.

Various extensions to standard MC simulation have been proposed which can improve performance in certain scenarios. For example, quasi-Monte Carlo (QMC) methods (Caflisch, 1998) use low-discrepancy sequences such as the Sobol sequence (Niederreiter, 1988) to sample from $p(\mathbf{X})$ more effectively, resulting in a convergence rate of approximately $\mathcal{O}(\log(n)^c/n)$, for some constant c. Markov Chain Monte Carlo (MCMC) methods increase the complexity of the algorithm by introducing dependence in the samples, and are particularly popular in Bayesian computations (Brooks et al., 2011). However, the computational expense required for good performance of all MC methods remains a large disadvantage, and a more sophisticated approach is required.

2.3. Surrogate methods

Driven by the need for a more computationally efficient solution to UQ problems in computer experiments, various surrogate modelling techniques have been proposed. A surrogate model — also known as a meta-model, an emulator or a response surface — acts as an approximation to the simulator. Built using information from the original simulator, it aims to imitate the behaviour of the simulator as closely as possible while being computationally faster to evaluate. Mathematically, the simulator $\eta(\mathbf{x})$ is approximated with a surrogate model, denoted $\hat{\eta}(\mathbf{x})$. Since $\hat{\eta}(\cdot)$ is a cheap version of $\eta(\cdot)$, it can be used in place of the original simulator to tackle the UQ objectives. Notably, the MC methods described in Section 2.2.2 can be easily implemented on the cheap surrogate. This introduces an extra source of uncertainty, that of replacing the simulator with a simpler model, meaning that any analysis using the surrogate will only be an estimate of that obtained with the original simulator. However, if the surrogate model is built accurately enough this discrepancy will be small, and can be quantified appropriately in the spirit of UQ (Stuart and Teckentrup, 2016).

Surrogate models may be built using non-intrusive or intrusive methods, or a combination of both. Non-intrusive methods build the surrogate model using only information contained in $\mathcal{D}$ and $\mathbf{y}$, that is, runs of the simulator at an experimental design. Intrusive methods construct the surrogate model by exploiting the equations that make up the mathematical component of the simulator. Surrogates that are built intrusively inherit physical properties of the simulator and hence can provide efficient and tailor-made UQ. However, the mathematical component of the simulator must be known in full, and the numerical code to implement it must be modified to build the surrogate. This can be computationally expensive and non-trivial — and is not possible for legacy codes or black-box simulators — and is the main drawback of intrusive methods. On the other hand, non-intrusive methods are

more flexible in that the simulator does not need to be modified in any way, only requiring "black-box" runs at an experimental design. Surrogates that are built non-intrusively do not explicitly inherit simulator properties, but it is often possible to encode knowledge of the simulator into the structure of the surrogate.

Many types of surrogate models exist in the literature, including: response surfaces or linear regression (Faraway, 2006), Gaussian process emulation or Kriging (Santner et al., 2003; Rasmussen and Williams, 2006), polynomial chaos (Ghanem and Spanos, 1991b; Xiu and Karniadakis, 2003), support vector machines (Steinwart and Christmann, 2008) and neural networks (Hagan et al., 1996). Gaussian process emulators and polynomial chaos are arguably the two most popular techniques for UQ in computer experiments and will be the main focus of this work. Their basic principles and development over time will now be discussed in detail in the two following sections.

2.3.1. Polynomial chaos

A popular surrogate method in the engineering and applied mathematics communities is polynomial chaos (PC). The term "polynomial chaos" (also known as homogeneous chaos) was coined by Wiener (1938), who studied spectral decompositions of random variables and Brownian motion, some time before (and unrelated to) the ideas of chaos theory in dynamical systems. Wiener (1938) proposed that any random variable could be represented as an infinite series expansion of Hermite polynomials evaluated in terms of Gaussian random variables. The Hermite polynomials are orthogonal with respect to the Gaussian probability distribution. This Fourier-like expansion became known as the polynomial chaos expansion (PCE), the Wiener-Hermite expansion, or Hermite-chaos. It was shown to be a special case of the Cameron-Martin theorem (Cameron and Martin, 1947), who proved that the Wiener-Hermite expansion converged in the mean-square for random variables with finite variance. Since the expansion must be truncated to finite order to be used in practice, a PCE can be viewed as a way of discretising a random variable or a random field into a finite number of polynomial basis functions evaluated at independent random variables.

A number of early applications of polynomial chaos can be found in the literature before their use in computer experiments. Firstly, Chorin (1971) demonstrated that the Wiener-Hermite expansion can be used to improve the quality of a Monte Carlo estimator, for example in calculating the numerical expectation of a function. By representing the function as a finite series of Hermite polynomials, the convergence of the Monte Carlo estimator can be accelerated and its variance can be reduced. These ideas were generalised to multiple dimensions by Maltz and Hitzl (1979) and Hitzl

and Maltz (1980). Secondly, Meecham and Jeng (1968) used a second-order Wiener-Hermite expansion to discretise both Gaussian and non-Gaussian random fields in turbulence and energy cascade problems. However, Crow and Canavan (1970) and Chorin (1974) indicated that such low-order representations were not suitable for capturing these typically high frequency processes. Furthermore, convergence for the non-Gaussian cases was very slow and higher order polynomial terms are required. Due to these complications, advancement of PC methods slowed and they were not used in practice until over a decade later.

Their resurgence and subsequent use in computer experiments was due to the seminal works of Ghanem and Spanos (1990, 1991b). They were primarily concerned with structural mechanics and engineering problems governed by known differential equations, in the presence of parameter uncertainty. Their main objective was to study how input uncertainty propagates through the computational model to the output process in an uncertainty analysis. The uncertain input is assumed to be a Gaussian process $Y(\mathbf{x})$ with zero mean and known covariance function, $\text{cov}(Y(\mathbf{x}), Y(\mathbf{x}')) = C(\mathbf{x}, \mathbf{x}')$ (for more information on Gaussian processes see Sections 2.3.2 and 3.3). As a dimension reduction tool, the random process is discretised using a Karhunen-Loève (KL) expansion (Loève, 1977):

$$Y(\mathbf{x}) = \sum_{k=0}^{\infty} \xi_k(\theta)\sqrt{\lambda_k} f_k(\mathbf{x}), \tag{2.4}$$

where λ_k and $f_k(\mathbf{x})$, $k = 0, \ldots, \infty$, are the eigenvalues and eigenfunctions from a spectral decomposition of the covariance function $C(\mathbf{x}, \mathbf{x}')$. Importantly, $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots)$ are independent and identically distributed standard Gaussian random variables which depend on parameters θ. In this way, the uncertain simulator inputs $\mathbf{x}$ have been transformed into another (infinite) set of uncertain variables $\boldsymbol{\xi}$. A similar decomposition of the output of the simulator, here denoted $u(\mathbf{x})$, is now required which deals with the propagation of input uncertainty through the simulator. Critically, the covariance structure of the output is not known a priori so a KL expansion cannot be used. Instead, a Wiener-Hermite PCE is used:

$$\begin{aligned} u(\mathbf{x}) = u(\boldsymbol{\xi}) = a_0 H_0 + \sum_{i_1=1}^{\infty} a_{i_1} H_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} a_{i_1 i_2} H_2(\xi_{i_1}, \xi_{i_2}) \\ + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \cdots, \end{aligned} \tag{2.5}$$

where $H_k(\xi_{i_1}, \ldots, \xi_{i_k})$ denote the Hermite polynomials of order k in terms of the standard Gaussian random variables from the KL expansion in Equation (2.4). The summation is performed over a multidimensional index $(i_1, \ldots, i_k) \subset \mathbb{N}^k$, satisfying $i_1 \geq \cdots \geq i_k$, $k \geq 1$, to allow the construction of multivariate polynomials of order

k. For example, $H_3(\xi_5, \xi_3, \xi_1)$ denotes the third order Hermite polynomial in terms of the variables ξ_5, ξ_3 and ξ_1. More information on Hermite polynomials, and the construction of multivariate polynomials, is given in Section 3.2.1.

The coefficients a_0 and $a_{i_1,\ldots,i_k}$ (satisfying the above conditions) are deterministic, but unknown and must be found using information from the simulator. To do this, Ghanem and Spanos (1990) use an intrusive method, which is as follows. The KL expansion in Equation (2.4) and PCE in Equation (2.5) are truncated to finite order and then substituted for the relevant quantities in the governing equations of the simulator. A Galerkin projection (Ghanem and Spanos, 1991b) is applied using orthogonality properties of the Hermite polynomials. This results in an expression for the coefficients of the PCE, which is a further coupled system of differential equations. Consequently, the original simulator must be modified and a suitable finite element solver applied to numerically estimate the values of the coefficients. After this is completed, the estimated coefficients are substituted into the truncated version of the PCE in Equation (2.5) to give the PC surrogate. Since the surrogate is just a polynomial function, it can be sampled cheaply for any configuration of its inputs, namely the set of independent Gaussian random variables. Hence, it can be used in a Monte Carlo fashion to estimate statistics of the simulator output, such as its probability density function, or simply predict the simulator output at a new input setting. Alternatively, Ghanem and Spanos (1990, 1991b) demonstrated how second-order statistics of the simulator output can be estimated directly from the PCE coefficients. For example, the mean of the simulator output is estimated as the first coefficient a_0.

This surrogate-based uncertainty quantification procedure is known as the stochastic finite element method (SFEM) in the literature. The concept of the SFEM is now standard in engineering and following the work of Ghanem and Spanos (1990, 1991b), has been effectively used for uncertainty analysis in many problems, including: nonlinear vibration (Ghanem and Spanos, 1991a); porous media (Ghanem, 1998; Ghanem and Dham, 1998); fluid flow (Le Maître et al., 2001, 2002; Xiu and Karniadakis, 2003); and seismic soil-structure (Ghiocel and Ghanem, 2002). In particular, these applications demonstrate a large computational gain over direct Monte Carlo sampling of the simulator.

The early uses of polynomial chaos in computer experiments involve representing the simulator output as Wiener-Hermite expansion, regardless of whether the simulator inputs are Gaussian distributed. While good results can be achieved for non-Gaussian inputs, typically the approximation is poor and very high-order polynomial terms must be used in the expansion for reasonable accuracy. This then increases the complexity of the subsequent analysis as more coefficients need to be estimated. As a solution to this, generalised polynomial chaos (gPC) was proposed

by Xiu and Karniadakis (2002, 2003). In the gPC framework, the simulator output is represented as the following generalised PCE:

$$\begin{aligned} u(\mathbf{x}) = u(\boldsymbol{\xi}) = a_0 I_0 + \sum_{i_1=1}^{\infty} a_{i_1} I_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} a_{i_1 i_2} I_2(\xi_{i_1}, \xi_{i_2}) \\ + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} I_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \cdots , \end{aligned} \tag{2.6}$$

where the $I_k(\xi_{i_1}, \ldots, \xi_{i_k})$ are polynomials of order k taken from the Askey scheme (Askey and Wilson, 1985), and are orthogonal to the distributions of the random variables, $p(\boldsymbol{\xi})$. For notational convenience, Equation (2.6) is often presented as:

$$u(\boldsymbol{\xi}) = \sum_{\alpha=0}^{\infty} a_\alpha \psi_\alpha(\boldsymbol{\xi}), \tag{2.7}$$

where there is a one-to-one correspondence between the polynomials $I_k(\xi_{i_1}, \ldots, \xi_{i_k})$ and $\psi_\alpha(\boldsymbol{\xi})$, and the coefficients $a_{i_1,\ldots,i_k}$ and a_α. Equations (2.6) and (2.7) are sometimes referred to as Wiener-Askey expansions (Xiu and Karniadakis, 2002). This extension means that most common probability distributions for the inputs are incorporated into the framework with the use of different orthogonal polynomial families. Table 2.1 shows the "distribution-polynomial" pairs which are admissible in gPC. Notably, the original Wiener-Hermite expansion for Gaussian random variables is a subset of the Wiener-Askey expansion. Another important case is when the simulator inputs are uniformly distributed, leading to the use of Legendre polynomials in the PCE. The gPC framework also naturally allows for several inputs with different probability distributions, and in this case the multivariate polynomials in Equations (2.6) and (2.7) are made up of products of the relevant one-dimensional polynomial families. Similar to Cameron and Martin (1947), Xiu and Karniadakis (2002, 2003) demonstrated that the generalised PCE of any function with finite variance is convergent in the mean-square sense.

Table 2.1.: Correspondence of orthogonal polynomial families from the Askey scheme to random variable distributions in generalised polynomial chaos ($N \geq 0$ is a finite integer). Adapted from Xiu and Karniadakis (2003).

	Input distribution	Polynomial family	Support
Continuous	Gaussian	Hermite	$(-\infty, \infty)$
	Gamma	Laguerre	$[0, \infty)$
	Beta	Jacobi	$[a, b]$
	Uniform	Legendre	$[a, b]$
Discrete	Poisson	Charlier	$\{0, 1, \ldots\}$
	Binomial	Krawtchouk	$\{0, 1, \ldots, N\}$
	Negative binomial	Meixner	$\{0, 1, \ldots\}$
	Hypergeometric	Hahn	$\{0, 1, \ldots, N\}$

The majority of research up until this point — including the gPC methodology of Xiu and Karniadakis (2002, 2003) — estimated the coefficients of the PCE using the approach outlined in Ghanem and Spanos (1991b), that is, an intrusive calculation. For this to be possible, one must be able to write down the complete mathematical model making up the simulator, substitute the PCE into the appropriate places in the equations and solve the resulting coupled system. While this technique allows for extremely efficient surrogate-based UQ, it is not always possible. This may be due to the fact that the mathematical component of the simulator is too complex, only known in part, or not known at all. Furthermore, a modification of the computational component of the simulator may be non-trivial, or impossible in the case of legacy codes. The addition of UQ may also compromise the original simulator code. Due to the fact that many modern simulators are effectively "black-box" models, much of the recent developments in polynomial chaos have been for non-intrusive techniques. These approaches simply assume access to a finite number of black-box runs of the simulator, and fall into two categories.

The first non-intrusive approach to PC is called the pseudo-spectral method (Xiu, 2010) or non-intrusive spectral projection (NISP) (Le Maître et al., 2002; Reagan et al., 2003). In this case, projecting the simulator output against each of the polynomial basis functions in Equation (2.7) and employing orthogonality of the polynomials gives the following expression for the PC coefficients:

$$\begin{aligned} a_\alpha &= \frac{\langle u(\boldsymbol{\xi}), \psi_\alpha(\boldsymbol{\xi})\rangle}{\langle \psi_\alpha(\boldsymbol{\xi}), \psi_\alpha(\boldsymbol{\xi})\rangle} \\ &= \frac{\int_{\boldsymbol{\Xi}} u(\boldsymbol{\xi})\psi_\alpha(\boldsymbol{\xi})\rho(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}}{\int_{\boldsymbol{\Xi}} \psi_\alpha^2(\boldsymbol{\xi})\rho(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}}. \end{aligned} \tag{2.8}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, or expectation, taken with respect to the distribution of the random variables, $\rho(\boldsymbol{\xi})$, which has support $\boldsymbol{\Xi}$. In words, the PC coefficients are found as the inner product of the simulator output and the polynomial basis functions, divided by the inner product of the polynomial basis functions with themselves. The denominator in Equation (2.8) is known and depends on the choice of polynomial family. In the non-intrusive setting, there is not an analytical solution to the numerator in Equation (2.8) since the simulator output $u(\boldsymbol{\xi})$ is not known until it is evaluated at a particular input configuration $\boldsymbol{\xi}^{(i)}$. Consequently, a numerical integration method must be used to estimate the integral, and hence the coefficients of the PCE. Several numerical integration techniques have been used in the literature, and generally rely on sampling or quadrature techniques.

In terms of sampling, a simple procedure was given by Ghanem et al. (2000), who suggested taking a random sample from $\rho(\boldsymbol{\xi})$ and then computing a statistical average of the corresponding integrand values in a Monte Carlo integration. This idea was improved upon by Ghiocel and Ghanem (2002), who used stratified sampling,

and Le Maître et al. (2002), Reagan et al. (2003) and Choi et al. (2004), who used Latin Hypercube sampling (McKay et al., 1979). Convergence of Monte Carlo integration is slow and may require a large number of expensive simulator runs for good accuracy, but this is dimension independent. The approach also lends itself naturally to parallel computing techniques.

An alternative to random sampling for estimating the integral in Equation (2.8) is to use a numerical quadrature rule. Here, the integral is approximated by computing a weighted average of integrand evaluations at a fixed number of specified input locations. Many choices of quadrature rule are available, but Gaussian quadrature rules are particularly suitable for PC, where the integration is taken with respect to a probability distribution. Moreover, an appropriate Gaussian quadrature rule is available for all distribution types in the gPC framework. For example, when $\rho(\boldsymbol{\xi})$ is a Gaussian distribution such that Hermite polynomials are used in the PCE, Gauss-Hermite quadrature is an optimal choice. In low to moderate input dimensions (five or less), a tensor product construction of one-dimensional Gaussian quadrature rules can be employed. In this case, the simulator is evaluated at a factorial or tensor grid design. This approach is used by Le Maître et al. (2002) and Debusschere et al. (2004) and is shown to give excellent agreement with direct Monte Carlo for a reduced computational cost. In higher dimensions, the size of the tensor grid design becomes very large so a different solution is required. Common practice is to employ a form of sparse grid quadrature based on a principle by Smolyak (Smolyak, 1963). In this formulation, a subset of the full tensor product quadrature points are sensibly chosen, reducing the number of simulator runs while retaining high approximation accuracy. This approach has been implemented successfully in a number of cases (Keese and Matthies, 2003; Knio and Le Maître, 2006; Xiu and Sherwin, 2007; Fichtl and Prinja, 2011). Recently, adaptive sparse grid quadrature algorithms have been presented (Conrad and Marzouk, 2013; Gilli et al., 2013; Winokur et al., 2013, 2016) which allow for sequential design of experiments. Another non-intrusive approach which is related to NISP is the method of stochastic collocation (Xiu and Hesthaven, 2005; Babuška et al., 2007; Nobile et al., 2008b,a). Recall that the idea of PC is to determine the expansion coefficients given a known set of polynomial basis functions. Stochastic collocation proceeds in the opposite fashion, by fixing the expansion coefficients to be known and constructing Lagrange interpolating polynomials for the basis. The key to the approach is the use of tensor or sparse grid designs of quadrature rules, giving the similarity to the NISP method (Eldred and Burkardt, 2009).

The second non-intrusive approach to PC is called the stochastic response surface method (SRSM) (Isukapalli et al., 1998), probabilistic collocation (Berveiller et al., 2004; Alkhatib and King, 2014), point collocation (Hosder et al., 2007), or more

simply, regression (Berveiller et al., 2006; Sudret, 2008; Blatman and Sudret, 2010a). In this framework, the PCE in Equation (2.7) is truncated to finite order and viewed as a linear regression model. The coefficients of the expansion are found by ordinary least squares, that is, the estimated coefficients are those which minimise the sum of the squared difference between the truncated PCE and the simulator output at a set of experimental design points. Much of the work in this area considers the optimal size and type of experimental design to ensure good estimation of the coefficients. With regards to the size of the design, it is essential that it is at least as large as the number of coefficients in the PCE for a well-determined linear system. Many authors suggest a design size double that of the number of coefficients (Isukapalli et al., 1998; Berveiller et al., 2004; Hosder et al., 2007), although this may change depending on the objective of the analysis. For example, Berveiller et al. (2006) suggest that a design size the same as the number of coefficients is adequate for prediction purposes, but many more than double is required for accurate estimation of the probability density function of the simulator output. In terms of the type of experimental design, early work used tensor grid designs comprising roots of polynomials (Isukapalli et al., 1998; Berveiller et al., 2004, 2006). However, more recent research has demonstrated that space-filling designs (for example Latin Hypercubes) can lead to better results (Hosder et al., 2007; Blatman and Sudret, 2010a).

Many extensions of classical linear regression are directly applicable to the non-intrusive regression approach for estimating the PC coefficients. This has lead to the development of so-called sparse PC methods which aim to reduce the number of coefficients in the PCE, retaining only those with large order of magnitude. This is important because in high input dimensions and for high order polynomial expansions, the number of coefficients to estimate becomes prohibitively large. Examples for effective basis selection in PC include: stepwise regression (Blatman and Sudret, 2010a); least angle regression with hyperbolic truncation sets (Blatman and Sudret, 2011); regression using l_1-minimisation (Jakeman et al., 2015); and compressive sensing (Doostan and Owhadi, 2011; Sargsyan et al., 2014).

In the context of uncertainty quantification, it is clear that the initial focus of PC was for efficient surrogate-based uncertainty analysis of an expensive simulator. However, that is not to say that PC has not been used for the other UQ objectives outlined in Section 2.2.1. Examples of sensitivity analysis using PC are extensive in the literature, with diverse applications such as reacting-flow simulations (Reagan et al., 2003), artery models (Xiu and Sherwin, 2007) and land surface simulators (Sargsyan et al., 2014). Using a PC surrogate, built using either using intrusive or non-intrusive methods, it is possible to analytically derive many sensitivity indices from the estimated coefficients in post-processing (Sudret, 2008; Blatman and Sudret, 2010b). These include the Sobol indices, or the main effects and interaction

terms from an ANOVA type decomposition. Furthermore, bootstrap sampling of the PC surrogate allows for the construction of confidence intervals for all of these indices. Examples of simulator calibration using observations of the physical system are more recent and are mostly concerned with fluid flow, ocean and climate problems (Mattern et al., 2012; Tagade and Choi, 2014; Sraj et al., 2016). The majority of cases assume a Bayesian framework for inverse problems, where measurements of the physical system are made with error, and related to a simulator via a model discrepancy term. A hierarchical Bayesian structure depending on hyperparameters allows for natural incorporation of uncertainty from the sources outlined in Section 2.2. MCMC sampling can be used to sample from the posterior distributions of the parameters but relies on repeated executions of an expensive simulator. To reduce the computational burden, Marzouk et al. (2007) and Marzouk and Najm (2009) suggested building a PC surrogate for the simulator using the intrusive techniques of Ghanem and Spanos (1991b) and Xiu and Karniadakis (2002). The PC surrogate is then combined with the MCMC sampling in the original Bayesian framework for inverse problems. Because intrusive calculations are not always possible, non-intrusive counterparts for Bayesian calibration were given in Berveiller et al. (2012), who used the regression technique, and Mattern et al. (2012), who used NISP with tensor product quadrature.

Finally, PCEs have been modified and applied to more specific problems in computer experiments. An important example is for multi-level or hierarchical simulators, where it is possible to run the computer model at different complexities. This may be due to differing spatial or temporal resolution, or more physical processes included in the mathematical component of the model. Crucially, more runs can be obtained for cruder versions of a simulator since it is faster to run. Polynomial chaos strategies for combining information from different levels of a simulator can be found in Narayan et al. (2014), who were mainly concerned with two-level simulators, and Teckentrup et al. (2015), who combined PCEs with ideas from Multi Level Monte Carlo.

It is important to note that the gPC framework of Xiu and Karniadakis (2002, 2003) represents the simulator output using a global polynomial basis. For simulator output exhibiting discontinuities or nonstationary behaviour as a function of its inputs, this approximation may not be appropriate. To rectify this, various local PC approaches have been developed including: PCEs using piecewise polynomials (Schwab and Todor, 2003; Babuška et al., 2004; Sargsyan et al., 2014) or wavelets (Le Maître et al., 2004a,b), and multi-element methods which decompose the random input space into subelements (Wan and Karniadakis, 2005; Jakeman et al., 2013).

It may also be the case that the distributions of the uncertain inputs of the simulator cannot be adequately represented with any of the distributions in a Wiener-Askey

expansion (see Table 2.1). A solution to this problem was presented by Soize and Ghanem (2004) which was termed arbitrary polynomial chaos. In this scheme, an arbitrary probability distribution can be placed on the inputs and an orthogonal polynomial basis derived to be used in a PCE. Applications of arbitrary PC can be found in Oladyshkin and Nowak (2012). Soize (2015) also developed PCE techniques suitable for multi-modal probability distributions.

To summarise, polynomial chaos is an effective surrogate modelling technique which represents the simulator output as a truncated series expansion of orthogonal polynomials evaluated at the uncertain inputs of the simulator. Their early use in computer experiments was mainly for engineering and fluid dynamics problems governed by known differential equations, such that an intrusive approach could be used to fit the surrogate. More recently, with the increase of black-box simulators, several non-intrusive alternatives have been proposed which either rely on numerical integration or linear regression. While the early focus was on the uncertainty analysis of expensive simulators, polynomial chaos has since become a popular surrogate-based tool for many UQ objectives in the applied mathematics and engineering communities.

2.3.2. Gaussian process emulation

An alternative surrogate method which has received much attention in the statistical community is Gaussian process emulation. A Gaussian process (GP) is an extension of the normal, or Gaussian, probability distribution. Instead of describing random variables that are scalars or vectors, a GP is a form of stochastic process which can model the properties of functions (Rasmussen and Williams, 2006). Similar to the mean and variance parameters of the Gaussian probability distribution, and the mean vector and covariance matrix of the multivariate Gaussian distribution, a GP is completely specified with a mean function and a covariance function. In general terms, the mean function controls the global trend of the function across its input space, whereas the covariance function governs the local behaviour (although the covariance function can also encode global properties). A feature of a GP is that its evaluation at any point in continuous space results in a Gaussian random variable. Furthermore, any finite set of such random variables have a multivariate Gaussian distribution (Rasmussen and Williams, 2006). While this may be too strong an assumption in some cases, the flexibility of the mean and covariance functions means that an extremely wide range of function behaviour is possible.

Gaussian process models were first applied to the field of computer experiments in the seminal paper of Sacks et al. (1989). They modelled the output of a deterministic

simulator, $y = \eta(\mathbf{x})$, as a realisation of a stochastic process, $Y(\mathbf{x})$, using the form:

$$Y(\mathbf{x}) = \sum_{i=1}^{q} \beta_i h_i(\mathbf{x}) + Z(\mathbf{x}). \tag{2.9}$$

In what would become common practice in the field, the stochastic process $Y(\mathbf{x})$ is decomposed into a regression model for the mean function, made up of q known basis functions, and a zero-mean Gaussian process $Z(\mathbf{x})$. In their examples, a simple constant trend is used for the regression model and the following covariance function family $C(\mathbf{x}, \mathbf{x}')$ is considered:

$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{j=1}^{d} \exp\left(-\lambda_j |x_j - x'_j|^p\right), \tag{2.10}$$

where $0 < p \leq 2$. This covariance family is stationary since it only depends on the distance between two d-dimensional input configurations $\mathbf{x}$ and $\mathbf{x}'$, and includes many common types of covariance functions with the variation of p. For example, the exponential covariance function is given with $p = 1$, and the squared exponential covariance function is given by setting $p = 2$ (resulting in an isotropic process). The parameter σ^2, known as the Gaussian process variance, controls the extent that the process can deviate from the mean function. The correlation length parameters λ_j, $j = 1, \ldots, d$, allow the smoothness of the process to vary in different input dimensions. More detail on the role of covariance functions for Gaussian processes is given in Section 3.3.

Given this initial set-up, Sacks et al. (1989) were primarily concerned with the prediction objective in Section 2.2.1. That is, given n simulator runs in the form $\mathbf{y} = \left(y^{(1)} = \eta(\mathbf{x}^{(1)}), \ldots, y^{(n)} = \eta(\mathbf{x}^{(n)})\right)$, how can we "best predict" the simulator output at an untried input setting $\mathbf{x}^*$? Their solution is taken from the established methodology of Kriging (Krige, 1951; Matheron, 1963; Stein, 1999) in geostatistics. Taking a classical frequentist standpoint, a best linear unbiased predictor (BLUP) is constructed by minimising the mean squared error between the predictor and the unknown function. The predictor is conditional on the hyperparameters β_i, $i = 1, \ldots, q$, σ^2 and λ_j, $j = 1, \ldots, d$, which are estimated using the method of maximum likelihood. The BLUP surrogate interpolates the simulator runs and can be used to predict the simulator output at an untried input configuration. As a by-product, the mean squared error acts as a measure of uncertainty. To reflect the deterministic nature of the simulator, this uncertainty is zero at the observed data but increases as we move away from them.

An alternative, Bayesian, interpretation of Gaussian process models in computer experiments is also possible. In this case, the model in Equation (2.9) is used to represent prior beliefs on the behaviour of the simulator. By conditioning on the

simulator runs contained in $\mathbf{y}$ using Bayes' rule, a posterior for the simulator can be derived which is also a Gaussian process. The updated mean and covariance functions of this process provide predictions as well as uncertainty at untried input configurations. The posterior Gaussian process surrogate is commonly called an *emulator*, especially in the Bayesian statistics literature (O'Hagan, 2006). Sacks et al. (1989) were clearly aware of this interpretation, and noted that the Bayesian alternative gives identical results to Kriging in the case of improper uniform priors for the regression hyperparameters. This Bayesian perspective was formally presented by Currin et al. (1991), who were also concerned with the prediction objective. Their approach also uses maximum likelihood to estimate the hyperparameters, but considers linear and cubic covariance functions, and can be seen as a natural extension of previous Bayesian interpolation methods (Kimeldorf and Wahba, 1970; Blight and Ott, 1975; O'Hagan, 1978).

In the context of UQ for computer experiments, the early works of Sacks et al. (1989) and Currin et al. (1991) only account for code uncertainty — where the output of the simulator for a given input setting is unknown until it is evaluated. The variance of the posterior Gaussian process quantifies this uncertainty given the information contained in the simulator runs. The Bayesian framework for Gaussian processes allows for natural incorporation of all types of uncertainty found in computer experiments (see Section 2.2). An important milestone in the development of GP methodology was the work of Kennedy and O'Hagan (2001), who were the first to attempt to include all forms of uncertainty in a unified approach. They were concerned with the calibration of an expensive simulator using observed data of the physical system, and proposed the following model:

$$z_i = \zeta(\mathbf{x}_i) + e_i = \tau\, \eta(\mathbf{x}_i, \boldsymbol{\theta}) + \delta(\mathbf{x}_i) + e_i. \tag{2.11}$$

Here, $\zeta(\cdot)$ denotes the true underlying physical system, which is observed as z_i with error e_i. The true process is decomposed into the expensive simulator $\eta(\cdot)$ multiplied by a regression parameter τ, and an independent discrepancy term $\delta(\cdot)$. A distinction is also made between known variable inputs $\mathbf{x}_i$ and unknown calibration inputs $\boldsymbol{\theta}$ of the simulator. Prior information on both the simulator and the discrepancy term is represented using a Gaussian process in a hierarchical fashion. Observational data and simulator runs are then used to construct posteriors of the simulator and discrepancy terms, and hence, the underlying physical system.

To derive the posteriors, Kennedy and O'Hagan (2001) use a similar technique to that of Currin et al. (1991). Namely, a improper uniform prior is placed on the regression parameters of the mean function, which allows them to be integrated out of the analysis analytically. A full Bayesian analysis would then proceed by also integrating out the hyperparameters of the covariance functions. However, this

is computationally challenging and would require a numerical procedure such as MCMC. Furthermore, this approach would demand a careful consideration of prior information for these hyperparameters, which is not obvious to specify. Instead, the hyperparameters are estimated numerically using maximum likelihood and treated as fixed, in what is known as a "plug-in" approach. Subsequent inference about the calibration parameters $\boldsymbol{\theta}$ uses conditional posterior distributions given the estimated values of the hyperparameters. As such, the methodology outlined in Kennedy and O'Hagan (2001) does not fully account for observation error, residual uncertainty, model discrepancy and code uncertainty. Nonetheless, it is argued that very little information is lost using the plug-in approach, for a large reduction in computational complexity and time. The so-called "Kennedy-O'Hagan" method is extremely popular in the literature and has become standard practice not only for probabilistic calibration of computer models, but for building the Gaussian process emulator itself. Examples of its application are in flyer plate experiments (Williams et al., 2006), nuclear energy (McFarland et al., 2007) and climate models (Guillas et al., 2009; Sansó and Forest, 2009), amongst many others.

Fully Bayesian implementations of Gaussian process emulators, which specify priors for all hyperparameters, can be found in: Neal (1998), who gave an overview of GP methods for regression and classification; Higdon et al. (2004, 2008); Ray et al. (2015), who were concerned with calibration (particularly using high-dimensional data); Qian and Wu (2008), who extended work by Kennedy and O'Hagan (2000) involving GP emulators for hierarchical or multi-level simulators; and Williamson and Blaker (2014), who developed dynamic GP emulators for time series output. Another important philosophy is that of Bayes linear emulation (Craig et al., 1996, 2001; Goldstein and Rougier, 2006), which differs from a fully Bayesian analysis by only requiring partial specification of probability information (the first two moments), rather than complete probability distributions.

Aside from prediction and calibration, Gaussian process methods have been developed to tackle other UQ objectives outlined in Section 2.2.1. Uncertainty analysis was first considered by Haylock and O'Hagan (1996), who analytically derived the distribution of the simulator output mean, as well as the mean and variance of the simulator output variance. Their results assume a GP emulator with squared exponential prior covariance structure, and a multivariate Gaussian distribution for the inputs of the simulator. Therefore, their methods are not generally applicable in practice. However, their Bayesian approach displayed better accuracy in estimating the mean of a iodine simulator using only 10 samples when compared to a traditional Monte Carlo method with 1000 samples. More detail was given in O'Hagan (1998), who considered the evaluation of the distribution function of the simulator output under similar idealised scenarios. A general solution to uncertainty analysis was

proposed by Oakley and O'Hagan (2002). They outlined an algorithm combining a GP emulator with Monte Carlo simulation to numerically estimate the distribution of any summmary of the simulator output, where analytical approaches would be intractable. The mathematical details of the Oakley and O'Hagan (2002) uncertainty analysis algorithm will be presented in Section 4.2.1.

The task of sensitivity analysis for expensive simulators using Gaussian process emulators was first considered by Welch et al. (1992). They used the methodology of Sacks et al. (1989) to perform a screening experiment, that is, to identify a subset of active inputs of the simulator. This was carried out by using a GP emulator to estimate the main effects (Saltelli et al., 2009) of each input in a variance-based global sensitivity analysis. O'Hagan (1998) studied analytical expressions for the mean and variance of the main effect and interaction terms for specific cases. This work was developed into a Bayesian framework for probabilistic sensitivity analysis by Oakley and O'Hagan (2004), similar to their previous treatment of uncertainty analysis in Oakley and O'Hagan (2002). They demonstrated how to effectively visualise and perform inference for many local and global sensitivity analysis indices, such as the main effects, interations terms and total sensitivity using a GP emulator of the simulator. An alternative method was also put forward by Reich et al. (2012), who first decomposed the simulator output using ANOVA techniques, and directly emulated the main effects terms using Gaussian processes.

An alternative to the probabilistic calibration approach of Kennedy and O'Hagan (2001) is history matching (Craig et al., 1996). Instead of finding the 'best input' setting of the simulator given observational data, history matching proceeds by ruling out regions of input space that would result in simulator output inconsistent with observed data. An advantage of this strategy is that it can be performed in so-called "waves", sequentially reducing the implausible input space for an effective management of computational resources. Originating in the oil industry (Craig et al., 1996), the method has since been applied to hydrocarbon (Craig et al., 2001), galaxy formation (Vernon et al., 2010) and climate (Williamson et al., 2013, 2015) models, and is usually combined with Bayes linear emulators.

Finally, like polynomial chaos, GP emulators have been adapted and applied to more specific problems in computer experiments. A Bayesian strategy for incorporating and combining information from several levels of a multi-level simulator was proposed by Kennedy and O'Hagan (2000), who used an autoregressive model relating multiple GP emulators. Their model was extended by Forrester et al. (2007) who used a Kriging framework, and Qian and Wu (2008) who implemented a fully Bayesian version using MCMC. More recent work on multi-level simulators include: Williamson et al. (2012), who gave a decision theoretic approach; Le Gratiet and Cannamela (2015), who proposed sequential design strategies; and Oughton and

Craig (2016), who examined cases where the different versions of the simulators are nested.

Another interesting case is where the assumption of stationarity — for example with the use of the covariance function in Equation (2.10) — is violated. If there is evidence that the statistical properties of the simulator output vary depending on the location in the input space, an appropriate modelling strategy is required. Many authors advise increasing the complexity of the mean function component to soak up the nonstationarity (Rougier et al., 2009a), but in some cases more advanced techniques are required. Nonstationary GP emulators have been put forward which involve spatial deformation (Sampson and Guttorp, 1992; Higdon et al., 1998; Schmidt and O'Hagan, 2003) and treed partitioning (Gramacy and Lee, 2008) of the input space. Ba and Joseph (2012) also suggest the use of two composite Gaussian processes, one accounting for the smooth global trend and one the local details.

It is also imperative to note that much of the GP emulation methodology mentioned so far — including the seminal works of Sacks et al. (1989) and Kennedy and O'Hagan (2001) — assume that the simulator output is a scalar quantity. In reality, many modern simulators generate multivariate output, for example in the form of a spatial field or time series. It may also be the requirement that multiple outputs of the simulator be emulated, taking into account the possible correlation between them. Multivariate or dynamic emulator approaches have been developed recently (Higdon et al., 2008; Rougier, 2008; Conti and O'Hagan, 2010; Fricker et al., 2013) to deal with these scenarios, either by reducing the dimensionality of the field output, including time or spatial location as additional inputs, or using nonseparable covariance structures.

To summarise, Gaussian process emulation is an effective surrogate modelling approach which treats the unknown simulator output as an realisation of a stochastic process. There are essentially two schools for building these types of surrogate models: a classical frequentist approach originating from the Kriging methodology in geostatistics; and a Bayesian approach that uses a Gaussian process to represent prior information on the simulator, which is then updated into a posterior conditional on simulator runs. The Bayesian approach is particularly popular in the statistics community and has been demonstrated to be a natural framework for incorporating many sources of uncertainty in computer experiments. While the early use of of GP emulators mainly considered the prediction objective, strategies for tackling other UQ objectives have been proposed and all show computational gains over Monte Carlo simulation.

2.3.3. Comparisons and hybrid approaches

As demonstrated in Sections 2.3.1 and 2.3.2, polynomial chaos and Gaussian process emulation have proved to be efficient surrogate-based tools for uncertainty quantification in the applied mathematics and statistics fields respectively. Their application to computer experiments and subsequent methodological development has largely taken place over the same time period, namely the last 25 years. While they are both essentially methods for tackling the same problems in UQ for expensive simulators, it is surprising that the two communities have mostly worked in parallel to one another with little or no collaboration. A subtle link can been drawn between the approaches in the work of Ghanem and Spanos (1991b), where the uncertainty in simulator inputs is modelled as a Gaussian random process, discretised using a Karhunen-Loève expansion and propagated through the computational model using a polynomial chaos expansion. However, the Gaussian random process is simply used as a representation of the uncertainty in a spatial field and does not relate to GP emulator as a surrogate model. Only recently has there been a concious effort to integrate the two surrogate communities by making them aware of one another, comparing the surrogate methods and possibly combining them in a hybrid approach. This work is still in its early stages but a small amount of research has been carried out, which will now be summarised.

Examples comparing PC and GP surrogate methods are few and far between in the literature. Typically, advancements in PC or GP are compared to traditional Monte Carlo simulation as a reference for a set of benchmark problems. One example of a indirect comparison can be found in Liu et al. (2017), who were concerned with the uncertainty analysis of a aerodynamical simulator with geometrically induced input uncertainty. They compared the accuracy of four non-intrusive surrogate approaches in estimating statistics of the simulator output. The surrogate approaches compared were radial basis functions, polynomial chaos using sparse grid quadrature, polynomial chaos using regression, and Kriging. Their performance in estimating the mean, standard deviation and other statistics of the output were compared to direct quasi-Monte Carlo sampling. Their analysis was further complicated by allowing the radial basis functions, PC regression and Kriging techniques to be gradient-enhanced, making use of derivative information from the simulator. Their results clearly show that gradient-enhanced surrogates are more efficient than the non-gradient-enhanced alternatives, as well as quasi-Monte Carlo simulation, but do not directly compare PC and GP.

Some thoughts on the comparative advantages and disadvantages of the two approaches were given by O'Hagan (2013). In the spirit of bringing the two communities closer together, O'Hagan (2013) provided a tutorial of polynomial chaos from a statistical perspective, as well as highlighting some drawbacks of the approach. The

intention was that a researcher from the applied mathematics community could do the same for Gaussian process emulation. However, the work was never published and this intention was never fulfilled. The main criticism of the PC methodology given concerned its treatment of uncertainty: whereas a posterior GP emulator is a full probabilistic interpretation of simulator given some data, accounting for code uncertainty where it has not yet been run, a PC surrogate simply gives a prediction with no uncertainty. Furthermore, the Bayesian framework for GP emulators allows for natural incorporation of all sources of uncertainty found in computer experiments, but for polynomial chaos the main focus is on parameter uncertainty. While this is partially correct, Bayesian frameworks including PC can be found in the literature and extend past just input uncertainty, for example Marzouk and Najm (2009).

The first example of a hybrid approach can be found in DiazDelaO and Adhikari (2009, 2011), who highlighted that the stochastic finite element method (Ghanem and Spanos, 1991b) is very expensive when the number of terms in the Karhunen-Loève expansion and the polynomial chaos expansion is large. This is because the size of the coupled linear system for the PC coefficients (derived through an intrusive calculation) becomes computationally prohibitive, as the number of finite elements needed grows exponentially. They proposed to build a GP emulator instead to act as a cheap approximation to the linear system. In this framework coupling PC and GP methods, is it not obvious where to place design points to train the GP emulator. The authors present an algorithm based on the Cholesky decomposition of the linear system as a solution to the design problem. Their method can be viewed as an effective way to reduce the computational cost of the intrusive class of PC approaches.

A non-intrusive hybrid approach was proposed by Schöbi et al. (2015). Their method — known as PC-Kriging — models the simulator $\eta(\mathbf{x})$ in a similar form as Sacks et al. (1989), that of a regression model plus a zero-mean Gaussian process (see Equation (2.9)). The main difference is that a truncated generalised PCE is used for the mean function, leading to the following model:

$$\eta(\mathbf{x}) = \sum_{\alpha=0}^{N} a_\alpha \psi_\alpha(\mathbf{x}) + Z(\mathbf{x}). \tag{2.12}$$

The PCE is truncated to include $N+1$ terms which aim to describe the global behaviour of the simulator. The zero-mean Gaussian process $Z(\mathbf{x})$ then deals with the local variability, and has covariance function $\text{cov}(\eta(\mathbf{x}), \eta(\mathbf{x}')) = C(\mathbf{x}, \mathbf{x}')$ which is left for the practitioner to choose. Using a set of simulator runs, the PC coefficients are found non-intrusively using the regression approach and the Gaussian process hyperparameters found using maximum likelihood. Since the number of coefficients

to estimate grows exponentially with the polynomial order and the input dimension, the sparse PC ideas of Blatman and Sudret (2011) are also used. A least angle regression algorithm (Efron et al., 2004) is used in combination with hyperbolic index truncation sets to perform a basis selection, retaining only the important terms in the PCE. In this sense, the method is entirely data-driven. PC-Kriging has been implemented successfully in a number of applications, including: the uncertainty and sensitivity analysis of a dosimetry model (Kersaudy et al., 2015); the reliability analysis of engineering problems (Schöbi and Sudret, 2015); and the estimation of rare events (Schöbi et al., 2016). A key conclusion of this work is that the PC-Kriging approach tends to perform better than, or at least as good as, PC or GP surrogates on their own.

2.4. Summary

Uncertainty quantification in computer experiments is an important and rapidly expanding field, applying to all areas of science which use simulators to conduct experiments about a physical phenomenon of interest. To draw meaningful and statistically valid conclusions from these experiments, it is crucial to incorporate the various sources of uncertainty which are present in the computer simulation framework. Traditional approaches for UQ in computer experiments, revolving around Monte Carlo simulation, rely on a large number of simulator runs for appreciable accuracy. This poses a problem for expensive simulators, where the completion of a single run may take a substantial amount of time. Driven by the need for computational efficiency, a more sophisticated solution involves replacing the expensive simulator with a cheap to evaluate approximation called a surrogate. Monte Carlo simulation can then instead proceed on the surrogate model, with the main limitation that this analysis can only ever be an approximation. However, surrogates can be built to accurately represent a simulator and the approximation uncertainty can be quantified. Two of the most popular surrogate methods are polynomial chaos and Gaussian process emulation, which originate in the applied mathematics and statistics communities respectively. While they have been shown to be effective surrogate-based UQ tools and have become state-of-the-art methods in their respective fields, very little research has been carried out to compare or possibly combine the two approaches. This is surprising considering they are both essentially methods to tackle the same problems in UQ and have been developed over roughly the same time period (the last 25 years). Recently, a number of hybrid approaches have been proposed but these are still in early development. Moreover, there is no doubt that other hybrid approaches combining the two methods may be possible. Direct comparisons of the two approaches are particularly lacking in the literature,

meaning that it is unclear in the UQ community which method may perform better in different modelling scenarios. There is much need for comparisons in terms of accuracy, flexibility, computational cost, and ease of implementation for a range of UQ objectives. Such a comparison would not only be useful for UQ practitioners in highlighting which method should be used in different cases, but would raise awareness of the two approaches and draw the UQ surrogate modelling community closer together. It is hoped that this thesis goes some way towards addressing these issues.

3. Surrogate modelling

In this chapter, the notation and methodology of the surrogate modelling techniques is given in the form that it will be used in the remainder of this work. The chapter is structured as follows. In Section 3.1, the notation of surrogate modelling in uncertainty quantification is introduced, along with some properties a surrogate model should possess. Methodology for the two surrogate modelling approaches used in this work, polynomial chaos and Gaussian process emulation, is then described in detail in Sections 3.2 and 3.3 respectively. At the end of both sections, demonstrative examples are given for simple one-dimensional and two-dimensional test functions (Sections 3.2.4 and 3.3.4). Finally, the chapter is concluded with some discussion in Section 3.4.

3.1. Introduction and notation

Consider a simulator $\eta(\cdot) : \mathbb{R}^d \to \mathbb{R}^r$ as a mapping from a d-dimensional set of inputs, $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, to a r-dimensional set of outputs, $\mathbf{y} = (y_1, \ldots, y_r) \in \mathbb{R}^r$. As described in Chapter 2, simulators are used in many areas of science to describe real world phenomena, where direct experimentation in the field may be expensive or impossible. Simulators are generally made up of two parts: a mathematical component and a computational component. The mathematical component of the simulator is a model typically comprising a system of complex ordinary or partial differential equations — or a hybrid combination of the two — in terms of a large number of inputs (d) and outputs (r). These equations aim to describe the physical system of interest using scientific models developed through theoretical and experimental studies. A key feature of the mathematical component of the simulator is that it cannot be solved analytically due to its inherent complexity. Therefore, it must be coupled with a computational component, which provides a means of numerically solving the equations of the mathematical model. Running the simulator at a particular input configuration $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ — numerically solving the mathematical model for this choice of inputs using the computational component — returns the corresponding simulator output, $\mathbf{y}^{(i)} = \left(y_1^{(i)}, \ldots, y_r^{(i)}\right) = \eta(\mathbf{x}^{(i)})$.

The complexity of both the mathematical component of the simulator, as well as

the computational procedure to numerically solve it, means that the evaluation of the output, $\mathbf{y}^{(i)}$, for a particular input, $\mathbf{x}^{(i)}$, may take a substantial amount of time and computing resources. In this respect, the simulator is referred to as being computationally expensive. Even though the equations of the mathematical model are known in full, their complexity, and the fact they have to be solved numerically, means that the simulator output for a specific choice of inputs is not really known until the simulator is run at that input setting. Furthermore, the evaluation of the simulator output may take hours, days, or even months to obtain. This leads to the common assumption that the simulator is a "black-box" (Kennedy and O'Hagan, 2001) — simply a function linking a set of inputs to a set of outputs. Many simulators are deterministic (Oakley, 2011), in the sense that each time the simulator is run at a specific input setting $\mathbf{x}^{(i)}$, the corresponding output $\mathbf{y}^{(i)}$ obtained is identical. The alternative to this is stochastic simulators (Mortensen and Haggerty, 1988), where the output $\mathbf{y}^{(i)}$ varies each time the simulator is run at the input $\mathbf{x}^{(i)}$. In this work, all simulators featured are deterministic and assumed to be black-box models.

In many cases, there is a single simulator output which is to be studied (often called a quantity of interest), such that $r = 1$ (Santner et al., 2003). This might be the raw simulator output itself or a function of it (such as the maximum, minimum or mean value of a time series or spatial field), and may represent a particular physical process or variable under consideration. In this case, $\mathbf{y} = y \in \mathbb{R}$, and the deterministic black-box simulator can be compactly written as $y = \eta(\mathbf{x})$. Single output simulators will be the focus of this work. When considering multiple simulator outputs at once, the complexity of the scientific analysis is increased since multiple input-output interactions must be taken into account (Conti and O'Hagan, 2010).

The practice of running a simulator at different input settings is known as a computer experiment (Santner et al., 2003). Due to the fact that the simulator is computationally expensive, a practitioner will only have the time or computing resources to perform a finite number (typically small), n, of simulator runs. In order to best learn about the physical process that the simulator is trying to describe, it is important to carefully select the input configurations at which to run the simulator. Throughout this work, the specific set of inputs at which the simulator is run will be referred to as the experimental design, and will be denoted $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$. This is not to be confused with the scientific field of the design of experiments (DoE) (Santner et al., 2003) — often itself called experimental design — where the optimal choice of input settings for a computer experiment is theoretically considered. The result of running the simulator at the experimental design, $\mathcal{D}$, is the corresponding set of output values, denoted $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(n)}\right)^\top$, obtained by evaluating $y^{(i)} = \eta(\mathbf{x}^{(i)})$ for $i = 1, \ldots, n$.

In uncertainty quantification (UQ) for computer experiments (Sullivan, 2015), the

information contained in $\mathcal{D}$ and $\mathbf{y}$ is used to complete a UQ objective, such as an uncertainty analysis or a calibration of the simulator. Traditional methods for UQ such as Monte Carlo simulation (Caflisch, 1998) are not feasible in the case of computationally expensive simulators, because the number of simulator runs available, n, is too small to obtain good accuracy. As explained in Section 2.3, a more efficient strategy is to use $\mathcal{D}$ and $\mathbf{y}$ to build a surrogate model, here denoted $\hat{\eta}(\cdot)$, which acts as an approximation to the simulator $\eta(\cdot)$. Mathematically this can be represented as $y = \eta(\mathbf{x}) \approx \hat{\eta}(\mathbf{x})$. Replacing the simulator with a surrogate is at the cost of a loss of accuracy in subsequent analyses, but the error can be made as small as desired with sufficient resources. More importantly however, in the spirit of UQ the uncertainty introduced by replacing the simulator with a surrogate can be quantified.

Surrogate models which are built using only the information contained in $\mathcal{D}$ and $\mathbf{y}$ are known as non-intrusive, as they require no knowledge of the equations which make up the mathematical component of the simulator, just the runs of a black-box function. On the other hand, intrusive surrogate methods make explicit use of the mathematical model of the simulator, and may have to alter the computational component in order to build the surrogate. All of the surrogate models featured in this work are non-intrusive because the simulators are assumed to be black-boxes.

A surrogate model should possess some, or all, of the following properties (each summarised by a keyword in parentheses):

- It should provide a fast approximation to the simulator at any untried input setting $\mathbf{x}^{(*)}$, at a fraction of the cost of the original simulator. In this way, the surrogate may be used in place of the simulator for subsequent UQ analyses in a Monte Carlo fashion (*fast*).

- It should reflect the deterministic nature of the simulator, in the sense that the surrogate should either exactly reproduce the outputs, $\mathbf{y}$, when evaluated at the original design $\mathcal{D}$, or reproduce them with a small amount of error (*deterministic*).

- It should provide uncertainty information about its predictions of the simulator output at a general input setting $\mathbf{x}^{(*)}$, accounting for code uncertainty (see the sources of uncertainty in outlined Section 2.2). A beneficial, but not necessary condition is for the surrogate to have no uncertainty at the original experimental design, $\mathcal{D}$, since the corresponding outputs, $\mathbf{y}$, are known and have been observed deterministically (*uncertainty*).

- It should provide a good quality approximation to the simulator, in that the predictions made by the surrogate should be close to the simulator for all input

settings in the domain of interest. If uncertainty information is also provided by the surrogate, this uncertainty should be well calibrated with the variation in simulator output. The concept of quality is problem dependent and can be measured in a number of ways (*quality*).

There are many types of surrogate model which vary in their complexity, functional form, and the scientific field in which they are used. As explained in Section 2.3, the focus of this work is on two of the most popular techniques in the UQ literature: polynomial chaos and Gaussian process emulation. The mathematical details of polynomial chaos and Gaussian process emulation will be presented in the following two sections.

3.2. Polynomial chaos

Polynomial chaos (PC) is a surrogate method originating in the engineering community (Ghanem and Spanos, 1990, 1991b), which represents the simulator output as a series expansion of polynomials in terms of the inputs. The exact form of this series expansion will be presented shortly in Section 3.2.1, but before this a number of modelling assumptions must be made and notation introduced. Essentially, PC provides a means of relating the simulator inputs, $\mathbf{x}$, to the simulator output, y. In particular, it is a method for propagating uncertainty in the inputs to the output. Recall the concept of parameter uncertainty outlined in Section 2.2 in Chapter 2, which concerns the uncertainty in specifying the input settings at which to run the simulator (since they may take a range of plausible values). The first step in building a PC surrogate is to quantify parameter uncertainty, that is, to make a probability statement about the values the simulator inputs may take. In the PC literature, this is done as follows (Xiu, 2009). The inputs, $\mathbf{x} = (x_1, \ldots, x_d)$, are modelled as a random vector $\mathbf{X} = (X_1, \ldots, X_d)$, with known joint probability density function (PDF) $f_{\mathbf{X}}(\mathbf{x})$ on a support $\mathcal{X}$ (quantifying parameter uncertainty). To use PC in practice, the further assumption is made that the inputs are independent of one another, so that their joint PDF may be written as:

$$f_{\mathbf{X}}(\mathbf{x}) = \prod_{j=1}^{d} f_{X_j}(x_j), \tag{3.1}$$

where $f_{X_j}(x_j)$ denotes the marginal PDF of input X_j, which has support $\mathcal{X}_j$. Analogously, the support $\mathcal{X}$ of $f_{\mathbf{X}}(\mathbf{x})$ can be split up into the individual supports $\mathcal{X}_j$ of $f_{X_j}(x_j)$:

$$\mathcal{X} = \prod_{j=1}^{d} \mathcal{X}_j. \tag{3.2}$$

For computer experiments, probability distributions for the inputs are usually assumed known or estimated through the process of expert elicitation (Kynn, 2008).

In the case of parameter dependence or a random field for the inputs, an extra decorrelation step is required before PC can be used. For example, the stochastic finite element method (Ghanem and Spanos, 1990) outlined in Section 2.3.1 considers simulators whose inputs were modelled as Gaussian random fields. The Karhunen-Loève expansion (Loève, 1977) was then used to discretise this random field into a set of uncorrelated standard Gaussian random variables for which PC could be employed. Other methods for transforming correlated random variables into uncorrelated ones were summarised in Eldred et al. (2008). These include applying the inverse Cholesky factor of the correlation matrix of the random variables (Eldred et al., 2008), or the use of Rosenblatt (Rosenblatt, 1952), Nataf (Der Kiureghian and Liu, 1986) and Box-Cox (Box and Cox, 1964) transformations. However, all the examples presented in this work involve independent random variables so these transformations will not be presented in detail here.

The uncertainty in the inputs propagates through the simulator, inducing uncertainty in the simulator output y. Under this formulation, the simulator output is now the random variable $Y \equiv \eta(\mathbf{X})$, with distribution $f_Y(y)$. In PC the primary interest is to study the distribution $f_Y(y)$, that is, perform an uncertainty analysis of the simulator. As will be demonstrated shortly in Section 3.2.1, this involves representing Y as a series expansion of polynomials in terms of the uncertain inputs $\mathbf{X}$. In particular, the distinguishing feature of PC is that polynomials in the expansion are chosen from well known orthogonal families, with the specific choice of polynomial family depending on the probability distributions of the inputs. The following section introduces orthogonality properties of polynomials and relevant polynomial families, before showing how they are implemented in a series expansion for the simulator output, known as the polynomial chaos expansion.

3.2.1. Choosing the polynomial basis

Orthogonal polynomials

Consider the set $\{\psi_k(X_j), k \in \mathbb{N}_0\}$ a family of univariate polynomials in terms of the random input X_j, where k denotes the degree of the polynomial. For example, $\psi_5(X_2)$ denotes a fifth order polynomial in terms of the random input X_2. A set of polynomials in X_j are said to be orthogonal with respect to a probability distribution $f_{X_j}(x_j)$, if the following inner product holds for all k, $l \in \mathbb{N}_0$:

$$\langle \psi_k(X_j), \psi_l(X_j) \rangle \equiv \int_{\mathcal{X}_j} \psi_k(x_j)\psi_l(x_j) f_{X_j}(x_j)\, \mathrm{d}x_j = \gamma_k^2 \delta_{kl}. \tag{3.3}$$

Notice that the inner product $\langle \psi_k(X_j), \psi_l(X_j) \rangle$ is equivalent to the expectation $\mathbb{E}\left[\psi_k(X_j)\psi_l(X_j)\right]$ taken with respect to the distribution $f_{X_j}(x_j)$. In Equation (3.3), δ_{kl} is the Kronecker delta, which takes the value 1 when $k = l$ and the value 0 when $k \neq l$. The normalisation constants, γ_k^2, are thus defined as:

$$\gamma_k^2 \equiv \int_{\mathcal{X}_j} \psi_k^2(x_j) f_{X_j}(x_j)\, \mathrm{d}x_j. \tag{3.4}$$

The normalisation constants are known for many classical orthogonal polynomial families (Chihara, 2011). Two well known orthogonal families which will be the primary focus of this work, the Legendre and the Hermite polynomials, are described in the following examples.

Example 3.2.1. (Legendre polynomials)
The Legendre polynomials are a classical family of orthogonal polynomials which are the solutions to Legendre's differential equation. Denoted $P_k(x)$ for $k \in \mathbb{N}_0$, the first few Legendre polynomials (up to degree $k = 5$) are $P_0(x) = 1$, $P_1(x) = x$, $P_2(x) = \frac{1}{2}(3x^2 - 1)$, $P_3(x) = \frac{1}{2}(5x^3 - 3x)$, $P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$ and $P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x)$. Given $P_0(x)$ and $P_1(x)$, the Legendre polynomials can also be defined recursively as (Chihara, 2011):

$$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x).$$

They are orthogonal with respect to the distribution $X \sim \mathrm{Unif}[-1, 1]$, that is, the uniform distribution with support $\mathcal{X} = [-1, 1]$. This distribution has PDF $f_X(x) = \frac{1}{2}$. The orthogonality condition from Equation (3.3) is as follows:

$$\langle P_k(X), P_l(X) \rangle = \int_{-1}^{1} P_k(x)P_l(x)\frac{1}{2}\,\mathrm{d}x = \frac{1}{2k+1}\delta_{kl}. \tag{3.5}$$

The normalisation constants $\gamma_k^2 = \int_{-1}^{1} P_k^2(x)\frac{1}{2}\,\mathrm{d}x = \frac{1}{2k+1}$. The Legendre polynomials for $k = 0, \ldots, 5$ are plotted in Figure 3.1.

Example 3.2.2. (Hermite polynomials)
The Hermite polynomials are another classical family of orthogonal polynomials. Denoted $H_k(x)$ for $k \in \mathbb{N}_0$, the first few (probabilists') Hermite polynomials (up to degree $k = 5$) are $H_0(x) = 1$, $H_1(x) = x$, $H_2(x) = x^2 - 1$, $H_3(x) = x^3 - 3x$, $H_4(x) = x^4 - 6x^2 + 3$ and $H_5(x) = x^5 - 10x^3 + 15x$. Given $H_0(x)$, the Hermite polynomials can also be defined recursively as (Chihara, 2011):

$$H_{k+1}(x) = xH_k(x) - \frac{d}{dx}\left[H_k(x)\right].$$

They are orthogonal with respect to the distribution $X \sim N(0,1)$, that is, the standard Gaussian distribution with support $\mathcal{X} = (-\infty, \infty)$. This distribution has PDF $f_X(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$. The orthogonality condition from Equation (3.3) is as follows:

$$\langle H_k(X), H_l(X) \rangle = \int_{-\infty}^{\infty} H_k(x) H_l(x) \frac{e^{-x^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x = k! \delta_{kl}. \tag{3.6}$$

The normalisation constants $\gamma_k^2 = \int_{-\infty}^{\infty} H_k^2(x) \frac{e^{-x^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x = k!$. The Hermite polynomials for $k = 0, \ldots, 5$ are plotted in Figure 3.2.

The Legendre and Hermite polynomial families will be the only ones used in this work. Other examples include the Laguerre polynomials, which are orthogonal with respect to the Gamma probability distribution on the range $[0, \infty)$, and the Jacobi polynomials, which are orthogonal with respect to the Beta probability distribution on the range $[-1, 1]$ (Chihara, 2011).

Orthogonality of univariate families of polynomials can easily be extended to the multivariate case with d inputs $\mathbf{x} = (x_1, \ldots, x_d)$. Assuming independence of the inputs, given by Equations (3.1) and (3.2), a multivariate polynomial can be constructed using a product of univariate polynomials in each of the d inputs:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = \psi_{\alpha_1}(x_1) \times \cdots \times \psi_{\alpha_d}(x_d), \tag{3.7}$$

where the vector $\boldsymbol{\alpha} \equiv (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ is an index defining the degree of each univariate polynomial. As an example, $\psi_{(2,4,3)}(\mathbf{x})$ corresponds to the multivariate polynomial $\psi_2(x_1)\psi_4(x_2)\psi_3(x_3)$, that is, a second order polynomial in x_1, a fourth order polynomial in x_2, and a third order polynomial in x_3. Recalling that $\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x})$, the multivariate version of the orthogonality condition in Equation (3.3) is given by the following inner product, which should hold for all $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}_0^d$:

$$\langle \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \psi_{\boldsymbol{\beta}}(\mathbf{X}) \rangle \equiv \int_{\mathcal{X}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}) \psi_{\boldsymbol{\beta}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \gamma_{\boldsymbol{\alpha}}^2 \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}, \tag{3.8}$$

corresponding to the expectation $\mathbb{E}\left[\psi_{\boldsymbol{\alpha}}(\mathbf{X})\psi_{\boldsymbol{\beta}}(\mathbf{X})\right]$ taken with respect to the distribution $f_{\mathbf{X}}(\mathbf{x})$. The Kronecker delta, $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$, takes the value 1 when $\boldsymbol{\alpha} = \boldsymbol{\beta}$ ($\alpha_1 = \beta_1, \ldots, \alpha_d = \beta_d$) and the value 0 otherwise. The normalisation constants, $\gamma_{\boldsymbol{\alpha}}^2$, are thus defined as:

$$\gamma_{\boldsymbol{\alpha}}^2 \equiv \int_{\mathcal{X}} \psi_{\boldsymbol{\alpha}}^2(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{3.9}$$

Due to the assumption of independence of the inputs, given by Equations (3.1) and (3.2), the d-dimensional integral in Equation (3.8) separates into the product of d one-dimensional integrals. Moreover, the normalisation constant satisfies $\gamma_{\boldsymbol{\alpha}}^2 =$

Figure 3.1.: The Legendre polynomials, $P_k(x)$, for $k = 0, \ldots, 5$.

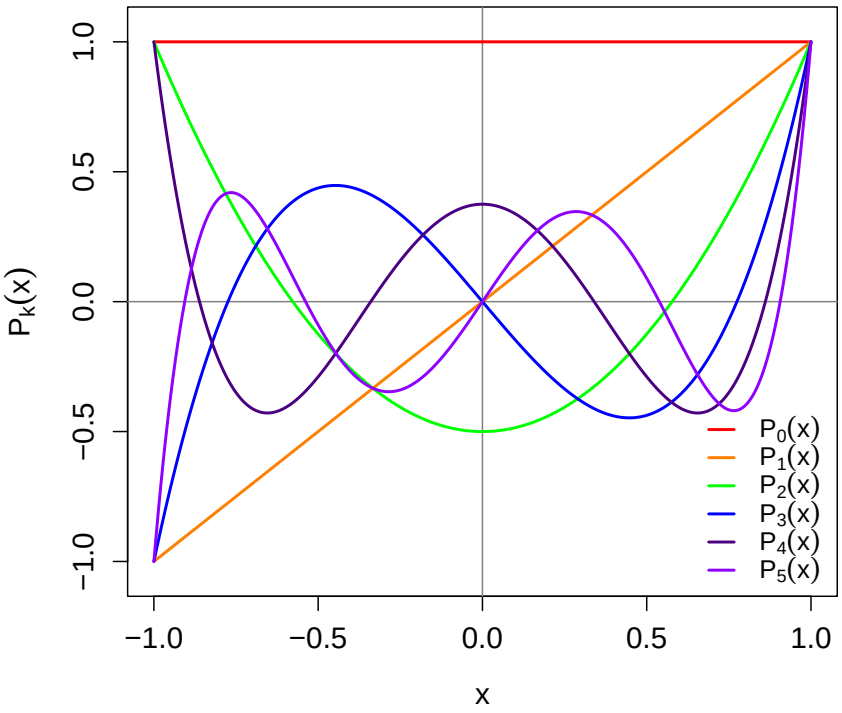
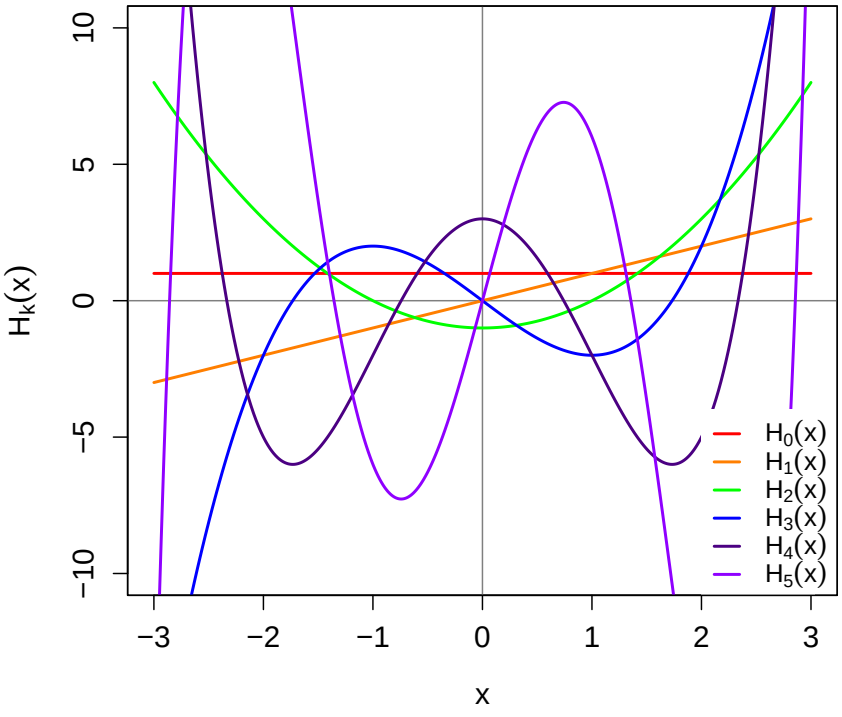


Figure 3.2.: The Hermite polynomials, $H_k(x)$, for $k = 0, \ldots, 5$.



$\gamma^2_{\alpha_1} \times \cdots \times \gamma^2_{\alpha_d}$, since:

$$\begin{aligned}\gamma^2_{\boldsymbol{\alpha}} &= \int_{\mathcal{X}} \psi^2_{\boldsymbol{\alpha}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \prod_{j=1}^{d} \left[\int_{\mathcal{X}_j} \psi^2_{\alpha_j}(x_j) f_{X_j}(x_j)\,\mathrm{d}x_j \right] \\ &= \gamma^2_{\alpha_1} \times \cdots \times \gamma^2_{\alpha_d},\end{aligned}$$

using the assumption of independence given by Equations (3.1) and (3.2), the construction of the multivariate polynomials in Equation (3.7), and the expression for the normalisation constants for one dimension in Equation (3.4). A specific example of a two-dimensional orthogonal polynomial, made up of both the Legendre and Hermite polynomial families, is now given for demonstrative purposes.

Example 3.2.3. (Orthogonality in 2-d)
Consider the case of two independent inputs $\mathbf{x} = (x_1, x_2)$ with $X_1 \sim \mathrm{Unif}[-1, 1]$ and $X_2 \sim N(0, 1)$. Their joint PDF is $f_{\mathbf{X}}(\mathbf{x}) = f_{X_1}(x_1) \times f_{X_2}(x_2) = \frac{1}{2} \times \frac{e^{-x^2/2}}{\sqrt{2\pi}}$. Using Legendre polynomials in x_1 and Hermite polynomials in x_2 gives the following form for the multivariate polynomials in $\mathbf{x}$:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = P_{\alpha_1}(x_1) \times H_{\alpha_2}(x_2). \tag{3.10}$$

Orthogonality of $\psi_{\boldsymbol{\alpha}}(\mathbf{x})$ is satisfied as:

$$\begin{aligned}\langle \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \psi_{\boldsymbol{\beta}}(\mathbf{X}) \rangle &= \int_{\mathcal{X}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}) \psi_{\boldsymbol{\beta}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{X}_1} \int_{\mathcal{X}_2} P_{\alpha_1}(x_1) H_{\alpha_2}(x_2) P_{\beta_1}(x_1) H_{\beta_2}(x_2) f_{X_1}(x_1) f_{X_2}(x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2 \\ &= \left(\int_{\mathcal{X}_1} P_{\alpha_1}(x_1) P_{\beta_1}(x_1) f_{X_1}(x_1)\,\mathrm{d}x_1 \right) \\ &\quad \times \left(\int_{\mathcal{X}_2} H_{\alpha_2}(x_2) H_{\beta_2}(x_2) f_{X_2}(x_2)\,\mathrm{d}x_2 \right) \\ &= \left(\int_{-1}^{1} P_{\alpha_1}(x_1) P_{\beta_1}(x_1) \frac{1}{2}\,\mathrm{d}x_1 \right) \\ &\quad \times \left(\int_{-\infty}^{\infty} H_{\alpha_2}(x_2) H_{\beta_2}(x_2) \frac{e^{-x^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x_2 \right) \\ &= \frac{1}{2\alpha_1 + 1} \delta_{\alpha_1 \beta_1} \times \alpha_2! \delta_{\alpha_2 \beta_2} = \gamma^2_{\boldsymbol{\alpha}} \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}},\end{aligned}$$

where the last step applies Equations (3.5) and (3.6), and the normalisation constant is $\gamma^2_{\boldsymbol{\alpha}} = \gamma^2_{\alpha_1} \times \gamma^2_{\alpha_2} = \frac{\alpha_2!}{2\alpha_1 + 1}$.

These examples demonstrate the intricate relationship between probability distribu-

tions and orthogonal polynomials, and provide all the tools required for representing the simulator output in a series expansion of polynomials. The particular form of series expansion used by the UQ community is known as the polynomial chaos expansion, which will now be described.

Polynomial chaos expansion

Assuming the uncertain simulator output Y is a second-order stationary process satisfying $\mathbb{E}[Y^2] < \infty$, it can be represented as the following series expansion (Xiu and Karniadakis, 2003):

$$Y = \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \qquad \mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x}). \tag{3.11}$$

The series in Equation (3.11) is known as a polynomial chaos expansion (PCE). It is an exact representation of Y, since the summation is over all values of the index $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, shown to converge in the mean-square sense (Xiu and Karniadakis, 2003).

In the PCE in Equation (3.11), $a_{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, are unknown constants which must be estimated using information from the simulator. There are various ways of doing this, which can be classified as intrusive or non-intrusive. Since the simulator is assumed to be a black-box, non-intrusive methods are the focus of this chapter, and two non-intrusive methods for estimating the coefficients will be presented later in Section 3.2.2.

More importantly for the construction of the PCE in Equation (3.11), $\psi_{\boldsymbol{\alpha}}(\mathbf{X})$, $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, are known multivariate orthogonal polynomials in terms of the random inputs $\mathbf{X}$, which are made up of the product of d univariate polynomials in each of the inputs x_j, $j = 1, \ldots, d$:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = \psi_{\alpha_1}(x_1) \times \cdots \times \psi_{\alpha_d}(x_d),$$

as already given in Equation (3.7). This is permitted due to the assumption of independence of the inputs, as given by Equations (3.1) and (3.2). The multivariate polynomials $\psi_{\boldsymbol{\alpha}}(\mathbf{X})$, $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, are chosen to be orthogonal with respect to the known joint probability distribution of the inputs, $f_{\mathbf{X}}(\mathbf{x})$. That is, the multivariate polynomials must satisfy the orthogonality condition in Equation (3.8). Again due to the assumption of independence, in practice this is done by selecting a univariate orthogonal polynomial family $\psi_k(x_j)$, $k \in \mathbb{N}_0$, for each input x_j, $j = 1, \ldots, d$, in turn. The appropriate polynomial family for the input x_j depends solely on the marginal probability distribution $f_{X_j}(x_j)$, as shown by the orthogonality condition in Equation (3.3). The multivariate orthogonal polynomials used in the PCE in Equa-

tion (3.11) are then simply the products of the appropriate univariate polynomials $\psi_{\alpha_j}(x_j)$, $j = 1, \ldots, d$, and the summation is performed over all possible combinations of the univariate polynomial degrees, $\alpha_j \in \mathbb{N}_0$, $j = 1, \ldots, d$. Consequently, the PCE can be written explicitly as:

$$Y = \sum_{\alpha_1=0}^{\infty} \cdots \sum_{\alpha_d=0}^{\infty} a_{(\alpha_1,\ldots,\alpha_d)} \psi_{(\alpha_1,\ldots,\alpha_d)}(\mathbf{X}).$$

As mentioned earlier, the specific form of the polynomials used in the PCE depends solely on the specification of the probability distributions for the inputs, $f_{X_j}(x_j)$, $j = 1, \ldots, d$. This is the framework of generalised polynomial chaos (gPC), first introduced by Xiu and Karniadakis (2003). In particular, the permitted orthogonal polynomials are taken from the Askey scheme (Askey and Wilson, 1985), and allow for most common probability distributions for the inputs. For this reason, the PCE in Equation (3.11) is sometimes referred to as a Wiener-Askey expansion (Xiu and Karniadakis, 2002). The full correspondence of orthogonal polynomial families from the Askey scheme to input probability distributions is presented in Table 2.1 in Chapter 2. Notice in particular that the Wiener-Hermite expansion developed by Wiener (1938) and Ghanem and Spanos (1991b) — where the inputs are assumed to be Gaussian so that the multivariate polynomials are made up of the product of univariate Hermite polynomials — is included as a special case of gPC. Another important case is where the inputs are assumed to be uniformly distributed, so that the multivariate polynomials are the products of univariate Legendre polynomials. However, it is not necessary for the inputs to have the same probability distributions. As long as they are independent, they can have different probability distributions, leading to a mix of orthogonal polynomial families being used in the PCE.

Despite the PCE in Equation (3.11) being an exact representation of the uncertain simulator output, Y, in terms of the inputs, $\mathbf{X}$, it is of little use in practice. In fact, for the PCE to be used as a surrogate for the simulator, the series must be truncated to finite order. There are a number of ways of doing this, which will now be described.

Truncating the polynomial chaos expansion

To motivate this section, consider the following example of a PCE for a simulator with two inputs.

Example 3.2.4. (A 2-d PCE)
Recall Example 3.2.3, where $\mathbf{x} = (x_1, x_2)$, $X_1 \sim \text{Unif}[-1, 1]$ and $X_2 \sim N(0, 1)$. Suppose these are two inputs to the simulator $Y = \eta(X_1, X_2)$. Employing orthogonality

with respect to these distributions gives a Legendre polynomial basis $P_k(X_1)$ in X_1 (Example 3.2.1) and a Hermite polynomial basis $H_k(X_2)$ in X_2 (Example 3.2.2). Following Example 3.2.3, a two-dimensional orthogonal polynomial is constructed as:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = P_{\alpha_1}(x_1) \times H_{\alpha_2}(x_2),$$

as already given in Equation (3.10), where the index $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$. A PCE representation of the uncertain simulator output Y is of the form:

$$\begin{aligned} Y &= \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^2} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}) \\ &= \sum_{\alpha_1=0}^{\infty} \sum_{\alpha_2=0}^{\infty} a_{(\alpha_1,\alpha_2)} P_{\alpha_1}(X_1) H_{\alpha_2}(X_2). \end{aligned} \tag{3.12}$$

Consider truncating the series in Equation (3.12) to second order, that is, up to and including polynomials of degree two. In this work, two possible ways of truncating the series will be featured. Firstly, the elements of the index $\boldsymbol{\alpha}$ can be constrained to sum to two or less, so that the truncated series is of the form:

$$\begin{aligned} Y &\approx \sum_{0 \leq \alpha_1 + \alpha_2 \leq 2} a_{(\alpha_1,\alpha_2)} P_{\alpha_1}(X_1) H_{\alpha_2}(X_2) \\ &= a_{(0,0)} P_0(X_1) H_0(X_2) + a_{(1,0)} P_1(X_1) H_0(X_2) + a_{(0,1)} P_0(X_1) H_1(X_2) \\ &+ a_{(2,0)} P_2(X_1) H_0(X_2) + a_{(1,1)} P_1(X_1) H_1(X_2) + a_{(0,2)} P_0(X_1) H_2(X_2). \end{aligned}$$

Secondly, all possible combinations of the univariate polynomials of degree zero to two can be included, so that the truncated series is of the form:

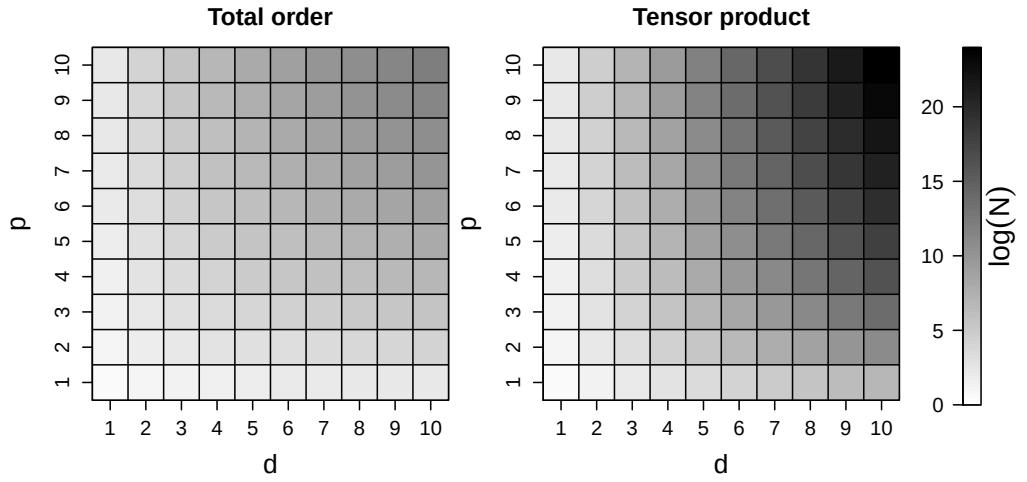
$$\begin{aligned} Y &\approx \sum_{\alpha_1=0}^{2} \sum_{\alpha_2=0}^{2} a_{(\alpha_1,\alpha_2)} P_{\alpha_1}(X_1) H_{\alpha_2}(X_2) \\ &= a_{(0,0)} P_0(X_1) H_0(X_2) + a_{(1,0)} P_1(X_1) H_0(X_2) + a_{(0,1)} P_0(X_1) H_1(X_2) \\ &+ a_{(2,0)} P_2(X_1) H_0(X_2) + a_{(1,1)} P_1(X_1) H_1(X_2) + a_{(0,2)} P_0(X_1) H_2(X_2) \\ &+ a_{(2,1)} P_2(X_1) H_1(X_2) + a_{(1,2)} P_1(X_1) H_2(X_2) + a_{(2,2)} P_2(X_1) H_2(X_2). \end{aligned}$$

The truncation of the PCE in Equation (3.11) will now be presented in general form. A PCE, truncated to finite order, p, will be denoted as:

$$Y \approx \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \qquad \mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x}), \tag{3.13}$$

where $|\boldsymbol{\alpha}|$ is a truncation scheme (Eldred and Burkardt, 2009) selecting the terms of the full PCE in Equation (3.11) to be retained. Clearly this is now only an ap-

Figure 3.3.: The logarithm of the number of terms in the truncated polynomial chaos expansion, N, as a function of the input dimension d and the polynomial order p, for total order (left panel) and tensor product (right panel) truncation schemes.



proximate representation of Y, which will become exact as $p \to \infty$. Applying such a truncation means that a finite set of polynomials are retained in the expansion, called the truncation set $\mathcal{A} = \{\psi_{\boldsymbol{\alpha}}(\mathbf{X}) : 0 \leq |\boldsymbol{\alpha}| \leq p\}$. The polynomials that are included in $\mathcal{A}$ depend on how $|\boldsymbol{\alpha}|$ is defined, and various truncation schemes are possible. As mentioned in Example 3.2.4, two truncation strategies will be considered. Firstly, the case of total order truncation (Eldred and Burkardt, 2009):

$$Y \approx \sum_{0 \leq \alpha_1 + \cdots + \alpha_d \leq p} a_{(\alpha_1,\ldots,\alpha_d)} \psi_{(\alpha_1,\ldots,\alpha_d)}(\mathbf{X}). \tag{3.14}$$

Total order truncation can be summarised as allowing all multivariate polynomials whose index $\boldsymbol{\alpha}$ sums to p or less to be included in the PCE. Secondly, the case of tensor product truncation (Eldred and Burkardt, 2009):

$$Y \approx \sum_{\alpha_1=0}^{p} \cdots \sum_{\alpha_d=0}^{p} a_{(\alpha_1,\ldots,\alpha_d)} \psi_{(\alpha_1,\ldots,\alpha_d)}(\mathbf{X}). \tag{3.15}$$

Tensor product truncation can be summarised as restricting the polynomial degree in each input dimension separately, and allowing all possible combinations of univariate polynomials to be included in the PCE.

The number of terms in the truncated PCE, here denoted N, is the cardinality of the truncation set: $N = |\mathcal{A}|$. For both the total order and tensor product truncation schemes, N grows rapidly with the number of inputs d and the truncation order p. Specifically, $N = \binom{d+p}{p}$ for a total order truncated expansion and $N = (p+1)^d$ for a tensor product truncated expansion (Eldred and Burkardt, 2009). A visualisation of the logarithm of N as a function of d and p is shown in Figure 3.3. It is evident that a tensor product truncation scheme retains many more terms in the PCE than

a total order truncation scheme, especially for large d and p. Since d is typically fixed for the simulator, the truncation order p must be chosen in accordance with computational restraints, that is, the number of simulator runs n that are available. As will be discussed in the next section, $n \geq N$ design points are sometimes required to estimate the PCE coefficients accurately.

With the chosen orthogonal polynomial basis and truncation scheme, the next step in PC is to estimate the unknown coefficients, $a_{\boldsymbol{\alpha}}$, $0 \leq |\boldsymbol{\alpha}| \leq p$, for the truncated PCE in Equation (3.13). This is performed using information from the simulator, and can be done in an intrusive (explicitly modifying the mathematical and computational components of the simulator) or non-intrusive (just relying on black-box runs of the simulator) manner. Since the assumption has been made that the simulator is a black-box, only non-intrusive methods are applicable because they make use of just information contained in $\mathcal{D}$ and $\mathbf{y}$. In this work, two of the most popular approaches for non-intrusively estimating the PC coefficients will be considered: regression (Blatman and Sudret, 2011) and non-intrusive spectral projection (Reagan et al., 2003). These will be described in the following section.

3.2.2. Finding the coefficients

Regression

The regression approach — also known as the stochastic response surface method (Isukapalli et al., 1998), probabilistic collocation (Berveiller et al., 2004), or point collocation (Hosder et al., 2007) — simply treats the truncated PCE in Equation (3.13) as a linear regression model. Here, the simulator output $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(n)}\right)$ at the experimental design $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$ is represented as:

$$y^{(i)} = \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}^{(i)}) + \epsilon^{(i)}, \qquad i = 1, \ldots, n, \tag{3.16}$$

where $\epsilon^{(i)}$ are scalar random variables accounting for the discrepancy between the simulator output and the truncated PCE. Let $\mathbf{a} = \{a_{\boldsymbol{\alpha}} : 0 \leq |\boldsymbol{\alpha}| \leq p\}$, that is, the set of PC coefficients to be estimated. Taking an ordinary least squares approach, the estimated coefficients, $\hat{\mathbf{a}}$, are found by minimising (Blatman and Sudret, 2011):

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a} \in \mathbb{R}^N} \left[\sum_{i=1}^{n} \left(\eta(\mathbf{x}^{(i)}) - \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}^{(i)}) \right)^2 \right]. \tag{3.17}$$

This can be interpreted as minimising the sum of the squared difference between the simulator output and the truncated PCE at the n experimental design points

contained in $\mathcal{D}$. The solution to Equation (3.17) is well known and reads:

$$\hat{\mathbf{a}} = \left(\psi^\top \psi\right)^{-1} \psi^\top \mathbf{y}, \tag{3.18}$$

where

$$\psi_{ij} = \psi_{\boldsymbol{\alpha}_j}(\mathbf{x}^{(i)}), \qquad i = 1, \ldots, n,\ j = 1, \ldots, N,$$

is a $n \times N$ model matrix containing the N polynomial terms of the truncated PCE evaluated at the n experimental design points. In this work, the regression approach is combined with the total order truncation scheme for PCEs, as is common in the literature (Eldred and Burkardt, 2009).

A key aspect of the regression approach for PC, just like the standard regression problem, is the size and type of the experimental design $\mathcal{D}$, which may affect the accuracy of the estimated coefficients. For the type of design, early work used tensor grid (factorial) designs comprising roots of polynomials (Isukapalli et al., 1998; Berveiller et al., 2004, 2006). However, more recent research has demonstrated that general space-filling designs, such as Latin Hypercube or Sobol sequence designs, lead to better results (Hosder et al., 2007; Blatman and Sudret, 2010a). This approach is also taken here, and in Chapter 4 Sobol sequence (Niederreiter, 1988) designs are used for this purpose. With regards to the size of design, n, critically it must be greater than or equal to the number of terms in the truncated PCE, N, to produce a well conditioned linear system. In Chapter 4, the rule of thumb $n = 2N$ is used, as suggested by many authors in the literature, for example Hosder et al. (2007). However, the regression technique is also implemented for $n = N$ in the numerical experiments in this work, to test the capability of the approach when faced with small design sizes.

Non-intrusive spectral projection

The second technique for estimating the coefficients is called non-intrusive spectral projection (NISP) (Le Maître et al., 2002; Reagan et al., 2003), also known as the pseudo-spectral method (Xiu, 2010). In this case, an equation for each of the PC coefficients can be derived in the following manner. Firstly, the truncated PCE in Equation (3.13) is multiplied by each polynomial basis term, $\psi_{\boldsymbol{\alpha}}(\mathbf{X})$, $0 \leq |\boldsymbol{\alpha}| \leq p$. This can be seen as projecting the output of the simulator to the space spanned by the chosen polynomial basis. Secondly, an inner product of the result is taken, and orthogonality of the polynomials employed using Equation (3.8). This procedure is known as a Galerkin projection (Eldred et al., 2008), and results in the following

expression for each coefficient:

$$\begin{aligned} a_{\boldsymbol{\alpha}} &= \frac{\langle Y, \psi_{\boldsymbol{\alpha}}(\mathbf{X})\rangle}{\langle \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \psi_{\boldsymbol{\alpha}}(\mathbf{X})\rangle} = \frac{\langle \eta(\mathbf{X}), \psi_{\boldsymbol{\alpha}}(\mathbf{X})\rangle}{\gamma_{\boldsymbol{\alpha}}^2} \\ &= \frac{1}{\gamma_{\boldsymbol{\alpha}}^2} \underbrace{\int_{\mathcal{X}} \eta(\mathbf{x})\psi_{\boldsymbol{\alpha}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, \mathrm{d}\mathbf{x}}_{\mathcal{I}_{\boldsymbol{\alpha}}}, \end{aligned} \tag{3.19}$$

where the expression for the simulator, $Y = \eta(\mathbf{X})$, as well as the orthogonality of the polynomials $\langle \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \psi_{\boldsymbol{\alpha}}(\mathbf{X})\rangle = \gamma_{\boldsymbol{\alpha}}^2$ (Equation (3.8)), has been used. For a truncated PCE with N terms, there are N d-dimensional integrals of the form $\mathcal{I}_{\boldsymbol{\alpha}}$ to solve. In a non-intrusive setting, these integrals cannot be written down, let alone be solved analytically, because the equations of the simulator, $\eta(\mathbf{x})$, are not known. Consequently, they must be solved using a numerical integration method.

A simple approach is to use sampling: generate a random sample from $f_{\mathbf{X}}(\mathbf{x})$ and compute a statistical average of the corresponding integrand values in a Monte Carlo integration. The method can also be improved upon with the use of quasi-random sampling (for example, Sobol sequences) or Latin Hypercube sampling. As explained in Section 2.2.2, the convergence of Monte Carlo methods is dimension independent but typically slow, such that a large number of simulator evaluations are needed for good accuracy. In the case of expensive simulators, these methods are therefore not appropriate for estimating the PC coefficients and will not be used in this work.

An alternative solution is to use numerical quadrature. There are several ways of constructing a quadrature rule for multidimensional integrals such as $\mathcal{I}_{\boldsymbol{\alpha}}$. In this work, a tensor product construction of one-dimensional quadrature rules in each input dimension is used. Denote a one-dimensional quadrature rule of n_j points for a function $g(x_j)$ in terms of input x_j as:

$$\mathcal{Q}^j\left[g(x_j)\right] \equiv \sum_{i=1}^{n_j} w_j^{(i)} g(x_j^{(i)}), \tag{3.20}$$

with weights $w_j^{(i)}$ and input settings $x_j^{(i)}$, $i = 1, \ldots, n_j$. A d-dimensional quadrature rule for a function $g(\mathbf{x})$ with $\mathbf{x} = (x_1, \ldots, x_d)$, can be constructed from a tensor product of one-dimensional quadrature rules, as follows:

$$\begin{aligned} \mathcal{Q}\left[g(\mathbf{x})\right] &\equiv \left(\mathcal{Q}^1 \otimes \cdots \otimes \mathcal{Q}^d\right)\left[g(\mathbf{x})\right] \\ &\equiv \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \left(w_1^{(i_1)} \times \cdots \times w_d^{(i_d)}\right) g(x_1^{(i_1)}, \ldots, x_d^{(i_d)}) \\ &\equiv \sum_{i=1}^{n} w^{(i)} g(\mathbf{x}^{(i)}), \end{aligned}$$

where $n = \prod_{j=1}^{d} n_j$. The weights $w^{(i)}$, $i = 1, \ldots, n$, are constructed by multiplying all possible combinations of the weights from the one-dimensional quadrature rules. Similarly, the input settings $\mathbf{x}^{(i)}$, $i = 1, \ldots, n$, are all possible combinations of the input settings from the one-dimensional quadrature rules. The resulting experimental design is then a tensor grid design. Applying the tensor product quadrature to Equation (3.19) gives the following estimate for the coefficients:

$$\begin{aligned}\hat{a}_{\boldsymbol{\alpha}} &= \frac{1}{\gamma_{\boldsymbol{\alpha}}^2} \mathcal{Q}\left[\eta(\mathbf{x})\psi_{\boldsymbol{\alpha}}(\mathbf{x})\right] \\ &= \frac{1}{\gamma_{\boldsymbol{\alpha}}^2} \sum_{i=1}^{n} w^{(i)} \eta(\mathbf{x}^{(i)}) \psi_{\boldsymbol{\alpha}}(\mathbf{x}^{(i)}). \end{aligned} \tag{3.21}$$

In this formulation, the user must choose one-dimensional quadrature rules, encompassing weights $w_j^{(i)}$ and input settings $x_j^{(i)}$, $i = 1, \ldots, n_j$, for each input dimension x_j. Several choices of quadrature rule are available. For the case of PC, the quadrature rules must be chosen to comply with the probability distribution $f_{X_j}(x_j)$ and support $\mathcal{X}_j$ of each input dimension, as these affect the form of the integral $\mathcal{I}_{\boldsymbol{\alpha}}$. In the gPC framework, there exists a suitable Gaussian quadrature rule for every possible probability distribution of the inputs (see Table 2.1). For example, when the simulator inputs have a Gaussian distribution, a Gauss-Hermite quadrature rule can be used, and when the inputs have a uniform distribution, a Gauss-Legendre quadrature rule can be used. This also aligns with the appropriate choice of polynomial basis in the gPC framework. Furthermore, when the simulator inputs have different probability distributions such that the polynomial basis is made up of a product of different polynomial families, the multidimensional quadrature rule in Equation (3.21) is a tensor product of different one-dimensional Gaussian quadrature rules.

In this work, the NISP method for evaluating the coefficients is combined with the tensor product truncation scheme for the PCE, as is common in the literature (Eldred and Burkardt, 2009). This is to align with the tensor product construction of the quadrature rule. Using a tensor product truncation, all combinations of one-dimensional polynomials up to degree p are included in the expansion. Therefore, $n_j = p + 1$ quadrature points are required in each dimension (since the zero order term $p = 0$ is included). This gives the total number of quadrature points as $n = (p+1)^d$. Since the size of the experimental design grows rapidly with the input dimension d, tensor product quadrature is only applicable to cases where $d \leq 5$ (Eldred et al., 2008). This is indeed the case for all examples featured in this work. In cases where $d > 5$, sparse grid quadrature is an alternative method which can reduce the computational burden while retaining high accuracy (Xiu and Hesthaven, 2005; Nobile et al., 2008b), or one must resort to sampling.

3.2.3. Polynomial chaos surrogate

The polynomial chaos surrogate, here denoted $\hat{\eta}^{PC}(\cdot)$, is given by substituting the estimated coefficients (estimated through regression or NISP), $\hat{a}_{\boldsymbol{\alpha}}$, into the truncated PCE from Equation (3.13):

$$Y = \eta(\mathbf{X}) \approx \hat{\eta}^{PC}(\mathbf{X}) \equiv \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} \hat{a}_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}). \tag{3.22}$$

Since this is just a polynomial function of the inputs, it can be evaluated very cheaply. Consequently, the simulator output at any new input setting $\mathbf{x}^{(*)}$ can be easily predicted as $\hat{\eta}^{PC}(\mathbf{x}^{(*)})$ without having to run the expensive simulator. This enables fast Monte Carlo (or a more efficient quadrature method) estimation of any function, $g(\cdot)$, of the uncertain simulator output, Y, as:

$$\mathbb{E}\left[g(Y)\right] \approx \frac{1}{m} \sum_{i=1}^{m} g(\hat{\eta}^{PC}(\mathbf{x}^{(i)})),$$

for a suitably large sample of size m from the input distribution $f_{\mathbf{X}}(\mathbf{x})$. In this way, the PC surrogate can be used to efficiently tackle the UQ objectives outlined in Section 2.2.1.

Statistical moments

Statistical properties of the simulator output Y can also be approximated directly from the PC coefficients as a post-processing step (Xiu, 2009). Using orthogonality properties of the polynomials, the expectation of Y is estimated as:

$$\mathbb{E}\left[Y\right] \approx \hat{a}_{\mathbf{0}}, \tag{3.23}$$

that is, the first (zero order) coefficient of the PCE. The variance of Y is approximated as:

$$\mathrm{var}[Y] \approx \sum_{1 \leq |\boldsymbol{\alpha}| \leq p} \hat{a}_{\boldsymbol{\alpha}}^2 \gamma_{\boldsymbol{\alpha}}^2, \tag{3.24}$$

where the summation is over all PCE coefficients except the zero order term. Higher order statistics and other quantities, such as sensitivity indices, can also be derived directly from the expansion (Ghanem and Spanos, 1991b), but these will not be featured here.

3.2.4. Examples

In this section, PC surrogates are applied to simple one-dimensional and two-dimensional examples. Firstly, consider the following simulator in a single input x:

$$y = \eta(x) = 5 + 5x + \sin(5 + 5x). \tag{3.25}$$

Suppose that the single input is uniformly distributed on the range $\mathcal{X} = [-1, 1]$, $X \sim \text{Unif}[-1, 1]$, and consider building a PC surrogate for the uncertain output $Y = \eta(X)$. As demonstrated in Section 3.2.1, a basis of Legendre polynomials should be used in the PCE since they are orthogonal to the uniform probability distribution. In this case, the truncated PCE in Equation (3.13) has the simplified form:

$$Y \approx \sum_{\alpha=0}^{p} a_\alpha P_\alpha(X)\,. \tag{3.26}$$

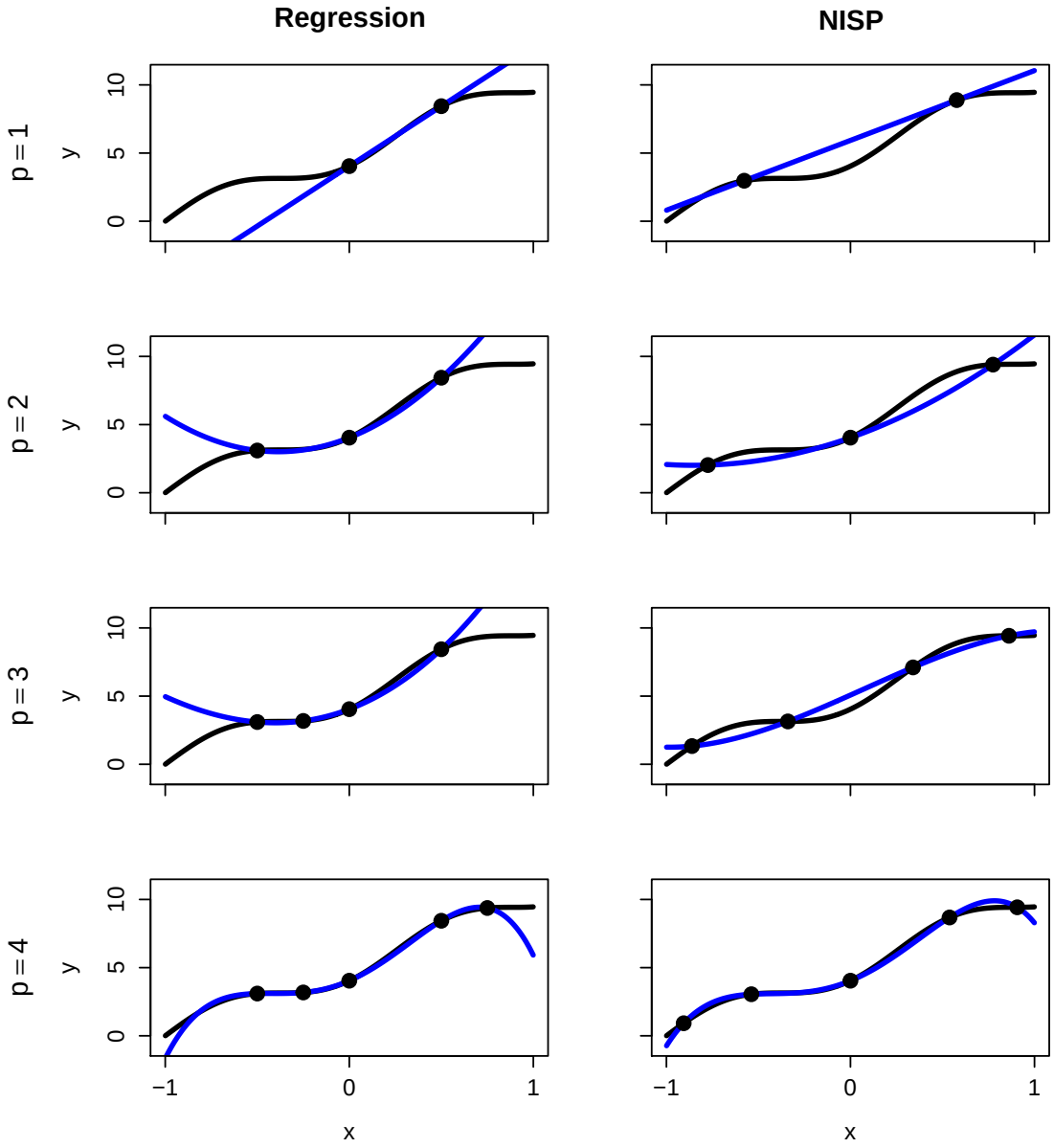
Note that the total order and tensor product truncation schemes are equivalent in one dimension. Furthermore, the number of terms in the expansion is $N = p + 1$. Consider fitting PC surrogates of orders $p = 1, 2, 3, 4$ using both the regression and NISP techniques for estimating the coefficients. For all cases, the design size is fixed at $n = N$, the minimum required. The type of experimental design for the regression cases are Sobol sequences, whereas Gauss-Legendre quadrature rules are implemented for the NISP approach.

Figure 3.4 shows the PC surrogates for the one-dimensional example in Equation (3.25). Clearly, as the truncation order is increased, the PC surrogate becomes closer to the function. However, more design points are required to evaluate higher order PC surrogates, since the number of coefficients to estimate is larger. For all cases, the PC surrogates interpolate the design points because of the fact that $n = N$. For $n > N$ this would not be the case. In general, the NISP PC surrogates are more accurate than the regression alternatives for a fixed design size; although this is largely due to the fact that the Gauss-Legendre quadrature design places points closer to the edges of the input design region. Conversely, the Sobol sequence design sequentially explores the design region from the middle outwards, and therefore leads to PC surrogates which are inaccurate at the edges. Even so, for the regression technique the design choice is completely arbitrary so this can be rectified easily. On the other hand, the Gauss-Legendre quadrature rule has fixed input settings, which cannot be changed unless an entirely different quadrature rule is employed.

Secondly, consider the following simulator in two inputs $\mathbf{x} = (x_1, x_2)$:

$$y = \eta(x_1, x_2) = \exp(-x_1)\tanh(5x_2). \tag{3.27}$$

Figure 3.4.: Polynomial chaos surrogates (blue) for the one-dimensional example $y = \eta(x) = 5 + 5x + \sin(5 + 5x)$ (black). The rows correspond to truncation orders $p = 1, 2, 3, 4$, which are built using design sizes $n = 2, 3, 4, 5$, and the columns refer to the regression and non-intrusive spectral projection (NISP) techniques for estimating the coefficients. Regression and NISP polynomial chaos surrogates are built on Sobol sequence and Gauss-Legendre quadrature rule designs respectively. The experimental design is shown as black dots in each case.



In this case, assume that the inputs are independent and uniformly distributed on the region $\mathcal{X} = [-1, 1]^2$, $X_1, X_2 \sim \text{Unif}[-1, 1]$, and consider building a PC surrogate for the uncertain output $Y = \eta(X_1, X_2)$. The assumption of independence is clearly justified here due to the separability of Equation (3.27), but this will not always be the case. Due to the uniform probability distribution and independence of the inputs, a multivariate polynomial is constructed using a product of Legendre polynomials in each input dimension:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{X}) = P_{\alpha_1}(X_1)P_{\alpha_2}(X_2). \tag{3.28}$$

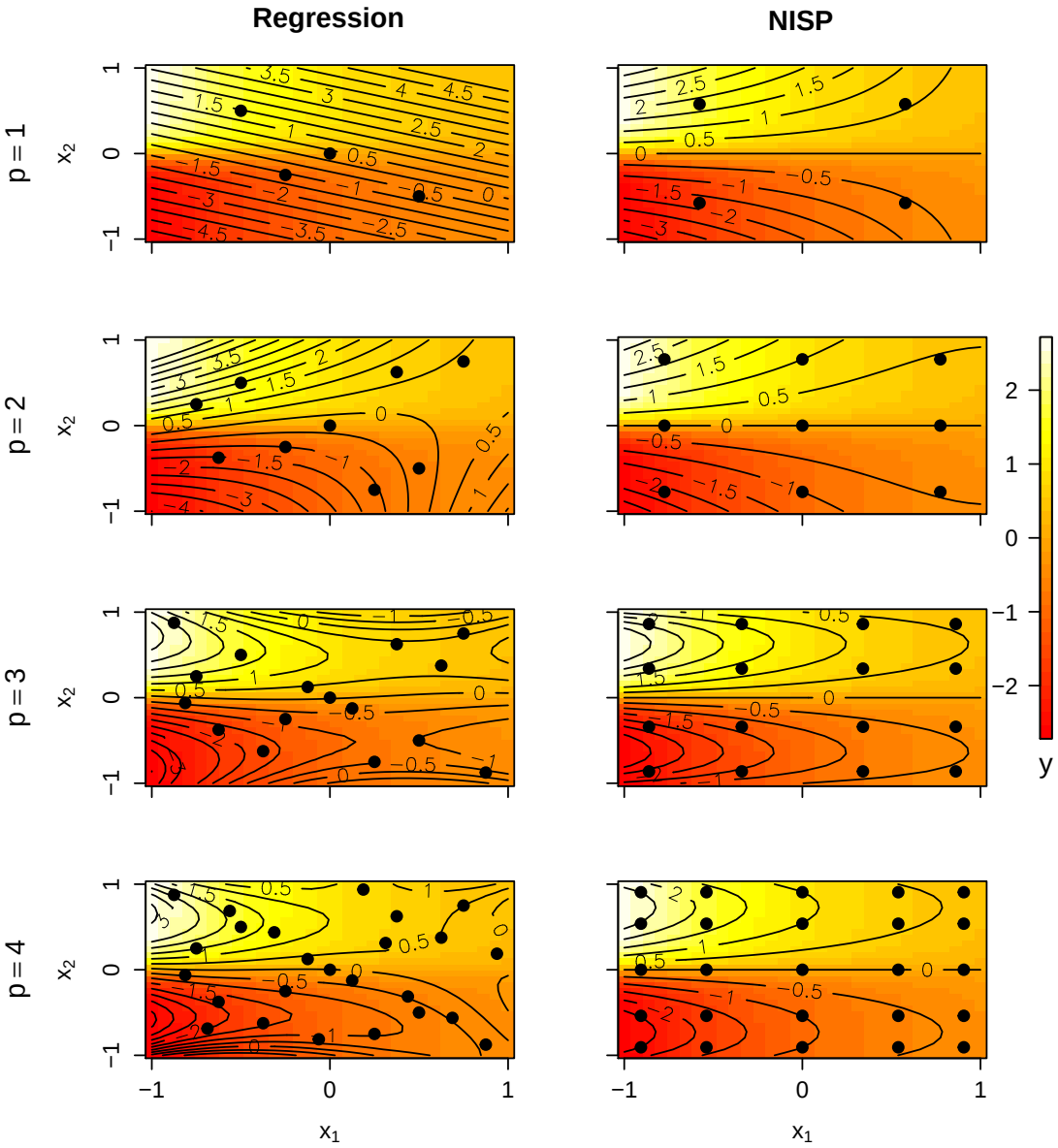
The truncated PCE then has the form:

$$Y \approx \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \tag{3.29}$$

where $\psi_{\boldsymbol{\alpha}}(\mathbf{X})$ is given by Equation (3.28). Since $d = 2$ here, a total order truncated PCE contains $N = \binom{2+p}{p}$ terms, and a tensor product truncated PCE contains $N = (p + 1)^2$ terms. Consider fitting PC surrogates of order $p = 1, 2, 3, 4$ using both the regression and NISP techniques for estimating the coefficients. As outlined in Section 3.2.1 the regression approach is combined with a total order truncation scheme, whereas tensor product truncation is used with the NISP technique. In this experiment, the design size is fixed at $n = N = (p+1)^2$ — the number of terms in the tensor product truncated PCE — for both non-intrusive approaches. This allows a fair comparison of the two non-intrusive approaches since the design size is the same for each truncation order. However, Sobol sequence and Gauss-Legendre quadrature rule design types are still used for the regression and NISP cases respectively. For the NISP method in particular, a tensor grid design is constructed using one-dimensional Gauss-Legendre quadrature rules in each input dimension.

Figure 3.5 shows the PC surrogates for the two-dimensional example in Equation (3.27). Again, as the truncation order is increased, the PC surrogate (shown by the black contour lines) becomes closer to the form of the function (shown by the coloured image). Clearly, the tensor product design of Gauss-Legendre quadrature rules, used for the NISP case, is more structured than the Sobol sequence designs. This leads to a more symmetrical appearance of the PC surrogates, which for this choice of function, is very accurate. On the other hand, the PC surrogates built using regression struggle to accurately reflect the nonlinear behaviour of the function. However, this is likely due to the fact that the NISP technique is combined with a tensor product truncation containing higher order interaction terms, which are required for this function.

Figure 3.5.: Polynomial chaos surrogates (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image). The rows correspond to truncation orders $p = 1, 2, 3, 4$, which are built using design sizes $n = 4, 9, 16, 25$, and the columns refer to the regression and non-intrusive spectral projection (NISP) techniques for estimating the coefficients. Regression and NISP polynomial chaos surrogates are built on Sobol sequence and tensor grid designs respectively. The experimental design is shown as black dots in each case.



3.3. Gaussian process emulation

Gaussian process (GP) emulation is a surrogate method originating from the Kriging methodology in geostatistics (Matheron, 1963) and subsequently developed by both the frequentist and Bayesian statistics communities. As outlined in Section 2.3.2, a GP is an extension of the Gaussian probability distribution, in that it is a type of stochastic process used to model the properties of functions, rather than scalars or vectors (Rasmussen and Williams, 2006). A GP is completely defined by its mean and covariance functions, similar to the mean and variance parameters in the Gaussian distribution[1]. In the case of computer experiments, the expensive simulator $\eta(\mathbf{x})$ is viewed as an unknown function of its inputs, and treated as a realisation of a GP. Mathematically, the GP assumption means that any collection of n simulator outputs $\mathbf{y} = \left(y^{(1)} = \eta(\mathbf{x}^{(1)}), \ldots, y^{(n)} = \eta(\mathbf{x}^{(n)})\right)$ can be modelled using a multivariate Gaussian distribution (denoted $MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$):

$$\mathbf{y} \sim MVN\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right). \tag{3.30}$$

The mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be fully specified by the practitioner or estimated in some way. In Sections 3.3.1 and 3.3.2, these quantities will take on structured forms and estimated using standard techniques from the GP literature.

In this work, a Bayesian approach to building a GP surrogate for the simulator is taken, which can be summarised as follows. Firstly, a GP is used as a prior distribution to describe any known behaviour of the simulator before it is run. This boils down to the choosing the form of the mean and covariance functions, and these can be made as complex as necessary to reflect any beliefs about the simulator. In the absence of any information, standard choices are also available. The task of setting up a GP prior is described in Section 3.3.1. With the prior specification complete, the simulator is run at the experimental design $\mathcal{D}$ and corresponding output values $\mathbf{y}$ observed. This information is then used in the second stage, namely updating the prior to a posterior distribution for the simulator. This is done using standard Bayesian methods, the mathematical details of which are presented in Section 3.3.2. The resulting posterior is a surrogate model for the simulator, known as an emulator in the GP literature (O'Hagan, 2006). The properties of the GP emulator, and examples of its application to simple functions, are given in Sections 3.3.3 and 3.3.4 respectively.

[1] A related strategy is that of Bayes linear statistics, which only requires partial specification of probability information through the first and second moments, rather than full distributional assumptions (Craig et al., 1996, 2001; Goldstein and Rougier, 2006).

3.3.1. Gaussian process prior

The behaviour of the simulator is modelled using the following Gaussian process prior:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(M(\mathbf{x}; \boldsymbol{\beta}), V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta})\right), \tag{3.31}$$

where $M(\mathbf{x}; \boldsymbol{\beta}) \equiv \mathbb{E}\left[\eta(\mathbf{x})\right]$ denotes the mean function, dependent on hyperparameters $\boldsymbol{\beta}$, and $V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}) \equiv \text{cov}\left[\eta(\mathbf{x}), \eta(\mathbf{x}')\right]$ denotes the covariance function, dependent on hyperparameters σ^2 and $\boldsymbol{\delta}$. The form of the mean and covariance functions considered in this work, as well as the definition and role of the hyperparameters, is outlined in the following sections.

Mean function

The mean function is used to describe any known global behaviour of the simulator output as a function of its inputs. As is common in the literature, the mean function is restricted to be a linear model in this work:

$$M(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta}, \tag{3.32}$$

where $\mathbf{h}(\mathbf{x})$ is a $q \times 1$ vector of known basis or regression functions, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_q)^\top$ are the corresponding coefficients to be estimated. The $\boldsymbol{\beta}$ hyperparameters are known as the mean function (or regression) hyperparameters. In the absence of any information about the simulator, it is common to use a constant mean function $M(\mathbf{x}; \boldsymbol{\beta}) = \beta_1$, where $\mathbf{h}(\mathbf{x})^\top = (1)$. Another popular choice is the linear mean function which allows for a linear trend in each input, $M(\mathbf{x}; \boldsymbol{\beta}) = \beta_1 + \beta_2 x_1 + \ldots + \beta_q x_d$, where $\mathbf{h}(\mathbf{x})^\top = (1\, x_1 \cdots x_d)$. A constant trend is used in the examples in this chapter.

For more complicated examples, such as those featured in Chapter 4, a slightly different strategy is used. In these cases, an initial mean function is chosen containing constant, linear and quadratic trends in each of the inputs, as well as first order interactions between them. Then, data from simulator runs is used in a preliminary regression analysis in conjunction with a stepwise selection algorithm to add and remove important terms using the Akaike information criterion (Faraway, 2006). The reduced model contains q basis functions and corresponding regression hyperparameters to estimate. This approach of fitting a complex mean function is common in the literature (for example Rougier et al. (2009a) and Vernon et al. (2010)), and can soak up any nonstationary behaviour of the simulator before introducing a covariance function.

Covariance function

In contrast to the mean function, the covariance function describes the local behaviour of the simulator. In particular, it describes how correlated the simulator output is at two input configurations $\mathbf{x}$ and $\mathbf{x}'$. In more general terms, the covariance function is chosen to reflect any known beliefs about the smoothness of the simulator output. In principle, any arbitrary covariance function can be used in specifying the GP prior. However, in practice a number of assumptions are typically made to simplify calculation and interpretation. Nonetheless, these assumptions still result in a GP prior which accounts for a wide range of simulator behaviour.

Firstly, as is common in the literature (Rasmussen and Williams, 2006), in this work stationarity is assumed, so that the covariance function can be written as:

$$V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}) = \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}), \tag{3.33}$$

that is, the product of a hyperparameter σ^2 known as the GP variance (which is constant across the input domain), and a correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) \equiv \text{corr}\left[\eta(\mathbf{x}), \eta(\mathbf{x}')\right]$, in terms of hyperparameters $\boldsymbol{\delta}$. The GP variance σ^2 determines the extent to which the GP can deviate away from the mean function, and the hyperparameters $\boldsymbol{\delta}$ control the strength of the correlation. Stationarity also restricts the correlation function to be a function of $\mathbf{x} - \mathbf{x}'$, $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = C(\mathbf{x} - \mathbf{x}'; \boldsymbol{\delta})$, meaning that it is invariant to translations in the input space (Rasmussen and Williams, 2006). In general terms, stationarity means that the correlation in simulator output does not depend on the position in the input space.

Secondly, in this work separability of the correlation function is assumed (Rasmussen and Williams, 2006), such that:

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \prod_{j=1}^{d} C(x_j, x_j'; \delta_j). \tag{3.34}$$

This means that the correlation in d-dimensional input space is equivalent to the product of the correlation in each of the inputs separately. In this framework, the hyperparameters $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_d) \in (0, \infty)^d$ are referred to as the correlation lengths, and control the strength of correlation in the simulator output in each of the inputs separately. In general, the larger the correlation length, the smoother the GP is in that input dimension.

In the absence of any information about how the simulator output may be correlated as a function of any two input settings, several standard and flexible choices for the correlation function are available. Since separable correlation functions are used here, the difference between $\mathbf{x}$ and $\mathbf{x}'$ can be defined separately in each input

dimension as $D_j = x_j - x'_j$, $j = 1, \ldots, d$. The main correlation function used in this work is the squared exponential (or Gaussian), which has the following form for d inputs (Rasmussen and Williams, 2006):

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\left(\frac{D_j}{\delta_j}\right)^2\right). \tag{3.35}$$

The squared exponential correlation function is infinitely differentiable, and thus results in a Gaussian process that is very smooth.

Another popular choice of correlation function is the Matérn family. This class of correlation function has an extra hyperparameter, ν, known as the shape parameter, which determines the diffentiability of the GP. Specifically, the GP is $\lfloor \nu \rfloor$ times differentiable (Rasmussen and Williams, 2006). Commonly, the shape hyperparameter is fixed at a half-integer to simplify the form of the correlation function. In this work, the Matérn correlation function is used with $\nu = 5/2$, which results in the following form:

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \left[\prod_{j=1}^{d}\left(1 + \sqrt{5}\left(\frac{|D_j|}{\delta_j}\right) + \frac{5}{3}\left(\frac{|D_j|}{\delta_j}\right)^2\right)\right]\exp\left(-\sqrt{5}\sum_{j=1}^{d}\left(\frac{|D_j|}{\delta_j}\right)\right). \tag{3.36}$$

This particular correlation function results in a GP that is twice differentiable, and consequently a less smooth process than that using a squared exponential correlation function with the same correlation lengths.

The squared exponential and the Matérn ($\nu = 5/2$) will be the only correlation functions featured in this work. They have been chosen due to the fact that they are most commonly used in the GP literature (Sacks et al., 1989; Kennedy and O'Hagan, 2001; Picheny and Ginsbourger, 2013). Other choices for the correlation function are the exponential, power exponential and Matérn with shape parameters set to other half-integer values (Rasmussen and Williams, 2006).

In summary, the full GP prior for the simulator used in this work is:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(M(\mathbf{x}; \boldsymbol{\beta}), \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\right), \tag{3.37}$$

where the mean function $M(\mathbf{x}; \boldsymbol{\beta})$ is a regression model as in Equation (3.32) and the correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$ is either the squared exponential or Matérn ($\nu = 5/2$) in Equations (3.35) and (3.36) respectively. With this choice of prior there are $q + d + 1$ hyperparameters which need to be estimated.

Examples

In this section, a short illustration of the effect of covariance function choice is given for a simple one-dimensional example. Consider placing the GP prior from Equation (3.37) on a simulator $\eta(x)$ which has a single input $x \in [0, 1]$. Without loss of generality, the mean function is chosen to be constant, $M(x, \boldsymbol{\beta}) = \beta_1$, with the single hyperparameter fixed at $\beta_1 = 0$. The covariance function is chosen to be the product of the GP variance σ^2 and a squared exponential correlation function with single correlation length hyperparameter δ.

Figure 3.6 shows five samples from this GP prior for four different choices for the GP variance σ^2 and correlation length δ hyperparameters. Prior uncertainty is represented as a region of two standard deviations around the prior mean, which in this case is $0 \pm 2\sigma$. For small choices of σ^2, the deviation from the mean value of zero is small, whereas for larger choices the deviation becomes larger. For small choices of δ, the GP is a highly varying function across the input range, but as this hyperparameter is increased the function becomes smoother. For all possible choices of the σ^2 and δ hyperparameters, any sample from the GP prior is an infinitely differentiable function due to the choice of squared exponential correlation function.

Figure 3.7 presents the same analysis, but for the Matérn correlation function in Equation (3.36). Similar conclusions can be drawn, but for the same hyperparameter choices it can be noted that the samples are slightly rougher functions than that of Figure 3.6. This is because the shape hyperparameter in the Matérn correlation function has been set to $\nu = 5/2$, so that each sample is a twice differentiable function (rather than infinitely differentiable). Values for the hyperparameters will not be known in most practical settings, and will have to be estimated from data in the process of deriving the posterior distribution for the simulator. This procedure will be described in the following section.

3.3.2. Deriving the posterior

With the prior specification complete and of the form in Equation (3.37), the second stage in building a GP surrogate is to condition on simulator runs to derive the posterior distribution. The mathematical steps necessary to derive the posterior will now be described. With the choices of mean and covariance function described in Section 3.3.1, the simulator outputs $\mathbf{y}$ can now be modelled as the following multivariate Gaussian distribution:

$$\mathbf{y} \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim MVN\left(\boldsymbol{\mu} = \mathbf{H}\boldsymbol{\beta}, \boldsymbol{\Sigma} = \sigma^2 \mathbf{A}\right), \tag{3.38}$$

Figure 3.6.: Five samples from one-dimensional Gaussian process priors (black lines) with squared exponential correlation functions, using different settings for the variance σ^2 and correlation length δ hyperparameters. The mean function is a constant $\beta_1 = 0$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sigma$ (grey shading), in each case.

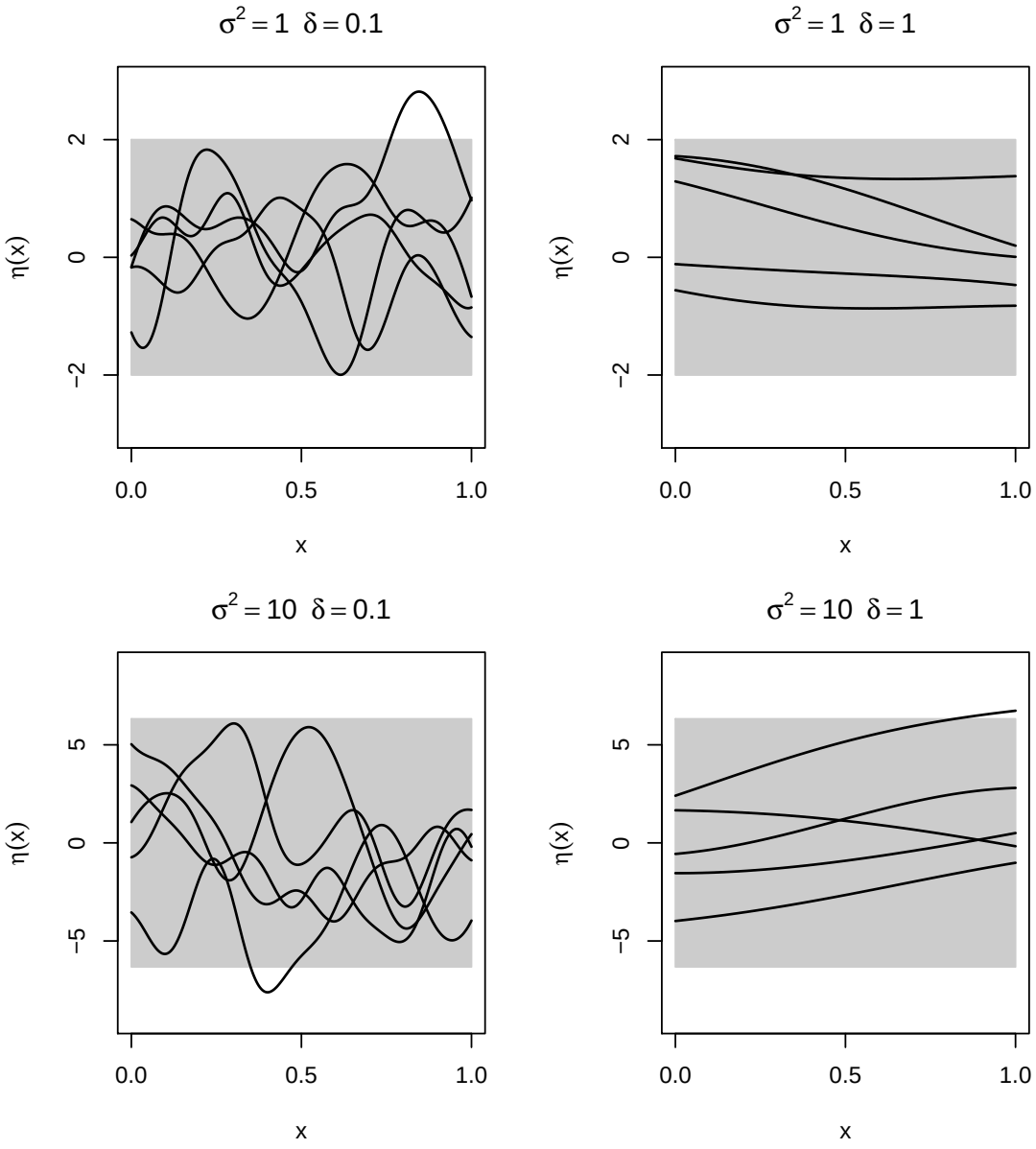
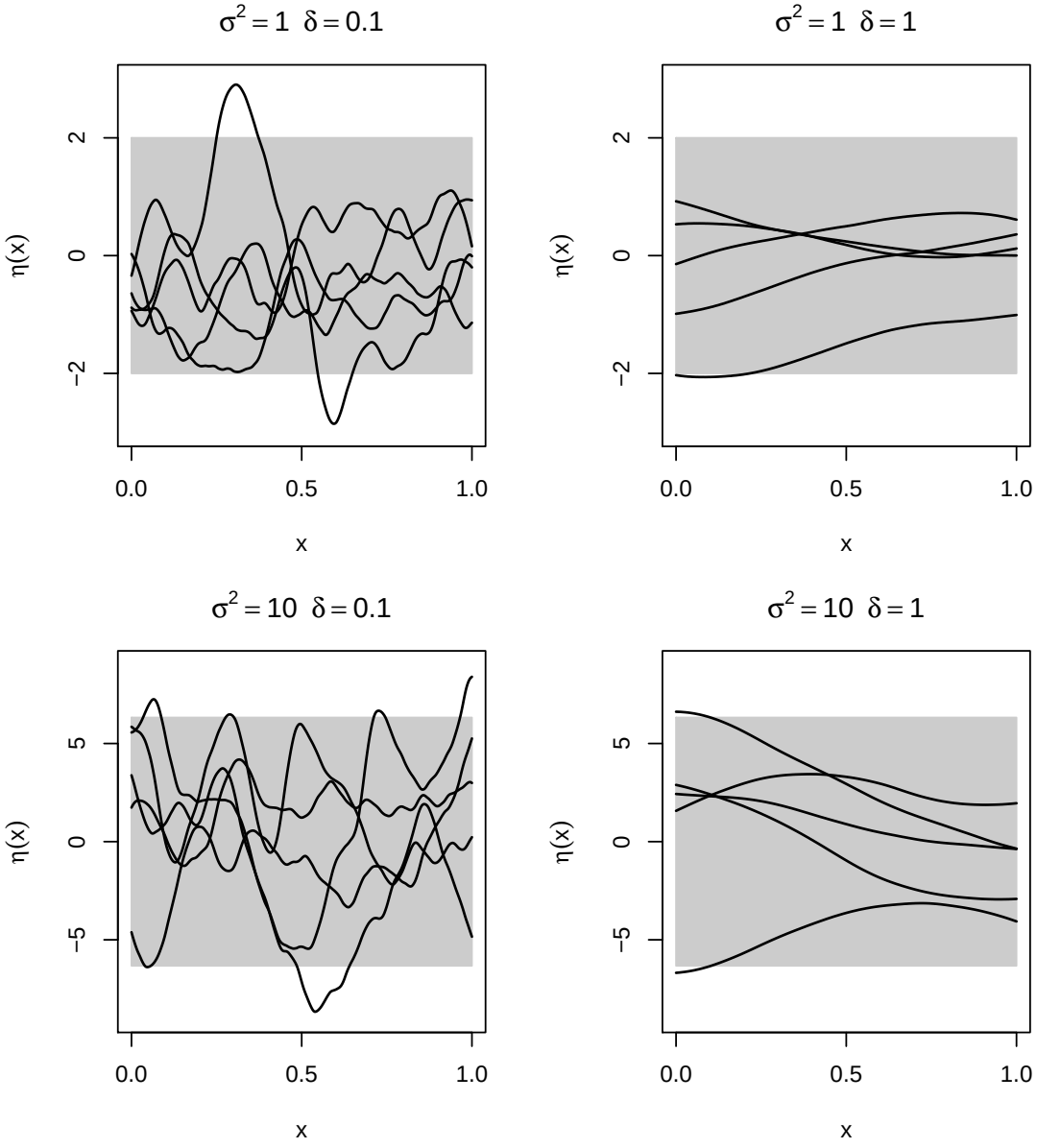


Figure 3.7.: Five samples from one-dimensional Gaussian process priors (black lines) with Matérn correlation functions (shape parameter $\nu = 5/2$), using different settings for the variance σ^2 and correlation length δ hyperparameters. The mean function is a constant $\beta_1 = 0$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sigma$ (grey shading), in each case.



where

$$\mathbf{H} = \left(\mathbf{h}(\mathbf{x}^{(1)})^\top, \ldots, \mathbf{h}(\mathbf{x}^{(n)})^\top\right)^\top, \tag{3.39}$$

that is, the q basis functions of the GP mean function evaluated at the n input points. Furthermore, $\mathbf{A}$ is an $n \times n$ matrix with elements:

$$\mathbf{A}_{i,j} = C(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}; \boldsymbol{\delta}), \quad i, j = 1, \ldots, n, \tag{3.40}$$

namely, the correlation function evaluated at all combinations of the n input points. Note that the dependence of $\mathbf{y}$ on the correlation lengths $\boldsymbol{\delta}$ comes implicitly through $\mathbf{A}$.

In the computer experiment, the simulator is evaluated at an experimental design $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$, giving the output $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(n)}\right)$. The next step is to combine the prior for the simulator in Equation (3.37) with the distribution in Equation (3.38) using Bayes' rule, to form a posterior for the simulator. Using standard techniques for conditioning in multivariate Gaussian distributions (Haylock and O'Hagan, 1996), the posterior GP for the simulator, conditional on the output values and the hyperparameters, is obtained as:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}, \mathbf{y} \sim \mathcal{GP}\left(M^*(\mathbf{x}; \boldsymbol{\beta}, \boldsymbol{\delta}), \sigma^2 C^*(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\right), \tag{3.41}$$

where

$$M^*(\mathbf{x}; \boldsymbol{\beta}, \boldsymbol{\delta}) = M(\mathbf{x}; \boldsymbol{\beta}) + \mathbf{t}(\mathbf{x})^\top \mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\boldsymbol{\beta}), \tag{3.42}$$

$$C^*(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) - \mathbf{t}(\mathbf{x})^\top \mathbf{A}^{-1} \mathbf{t}(\mathbf{x}'), \tag{3.43}$$

and

$$\mathbf{t}(\mathbf{x}) = \left(C(\mathbf{x}, \mathbf{x}^{(1)}; \boldsymbol{\delta}), \ldots, C(\mathbf{x}, \mathbf{x}^{(n)}; \boldsymbol{\delta})\right)^\top, \tag{3.44}$$

that is, the vector of correlations between the input setting $\mathbf{x}$ and the n experimental design points.

The posterior in Equation (3.41) is conditional on the hyperparameters. Since in a general setting it is a nontrivial task to specify values for $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$, the posterior distribution of $\eta(\mathbf{x}) \,|\, \mathbf{y}$ is desired. Thus, the hyperparameters need to be analytically integrated out of the posterior. In the absence of any knowledge on the mean function and variance hyperparameters, setting a weak prior $p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}$ allows $\boldsymbol{\beta}$ and σ^2 to be integrated out (Haylock and O'Hagan, 1996). The resulting posterior for the simulator, still conditional on the correlation lengths $\boldsymbol{\delta}$, is the following Student's-t distribution with $n - q$ degrees of freedom:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\delta}, \mathbf{y} \sim t_{n-q}\left(M^{**}(\mathbf{x}; \boldsymbol{\delta}), \hat{\sigma}^2 C^{**}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\right), \tag{3.45}$$

where

$$M^{**}(\mathbf{x};\boldsymbol{\delta}) = M^{*}(\mathbf{x};\hat{\boldsymbol{\beta}},\boldsymbol{\delta}), \tag{3.46}$$

$$C^{**}(\mathbf{x},\mathbf{x}';\boldsymbol{\delta}) = C^{*}(\mathbf{x},\mathbf{x}';\boldsymbol{\delta}) + \mathbf{Q}(\mathbf{x})(\mathbf{H}^{\top}\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{Q}(\mathbf{x}')^{\top}, \tag{3.47}$$

and

$$\mathbf{Q}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^{\top} - \mathbf{t}(\mathbf{x})^{\top}\mathbf{A}^{-1}\mathbf{H}.$$

The mean function and variance hyperparameters are as estimated as (Haylock and O'Hagan, 1996):

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^{\top}\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\top}\mathbf{A}^{-1}\mathbf{y}, \tag{3.48}$$

$$\hat{\sigma}^2 = \frac{1}{n-q-2}(\mathbf{y}-\mathbf{H}\hat{\boldsymbol{\beta}})^{\top}\mathbf{A}^{-1}(\mathbf{y}-\mathbf{H}\hat{\boldsymbol{\beta}}). \tag{3.49}$$

A fully Bayesian analysis would proceed by integrating out the correlation length parameters, $\boldsymbol{\delta}$, from the posterior in Equation (3.45). However, it is not possible to do this analytically, and a numerical procedure such as Markov Chain Monte Carlo would have to be used to this end, typically at a high computational cost. Moreover, a proper prior would have to be elicited for the correlation lengths, as otherwise the posterior for these hyperparameters would be improper (Kennedy and O'Hagan, 2001). The specification of a prior distribution for the correlation lengths is not a trivial task and would require careful consideration. For these reasons, a fully Bayesian analysis is rare in the literature. Instead, the approach of Kennedy and O'Hagan (2001) is followed in this work, where maximum a posteriori estimates of the correlation lengths are derived and then treated as known. In particular, the correlation lengths are estimated using maximum likelihood. The marginal likelihood for the correlation lengths given the data is (Andrianakis and Challenor, 2012):

$$\mathcal{L}(\boldsymbol{\delta}\,|\,\mathbf{y}) \propto (\hat{\sigma}^2)^{(n-q)/2}|\mathbf{A}|^{-1/2}|\mathbf{H}^{\top}\mathbf{A}^{-1}\mathbf{H}|^{-1/2}. \tag{3.50}$$

The correlation lengths are then estimated by maximising the logarithm of this likelihood:

$$\hat{\boldsymbol{\delta}} = \arg\max_{\boldsymbol{\delta}}(\log \mathcal{L}(\boldsymbol{\delta}\,|\,\mathbf{y})). \tag{3.51}$$

In this work, GP emulators are built using the `DiceKriging` (Roustant et al., 2012) package in the `R` (R Core Team, 2016) programming language. The package maximises the log-likelihood in (3.51) using the "L-BFGS-B" optimisation algorithm, which is a quasi-Newton scheme constrained by given lower and upper bounds (Byrd et al., 1995; Park and Baek, 2001). This algorithm also makes use of analytical gradients of the log-likelihood with respect to the correlation length parameters to speed up the optimisation. The number of iterations required for accurate estimation of the correlation lengths is problem dependent, but the main cost stems from repeatedly inverting the $n \times n$ correlation matrix $\mathbf{A}$ at a cost of $\mathcal{O}(n^3)$ each time. Furthermore, numerical issues in the inversion can arise when $\mathbf{A}$ becomes a near sin-

gular matrix. To alleviate these numerical problems, a small constant known as the nugget parameter is added to the diagonal elements of the correlation matrix in this work (Andrianakis and Challenor, 2012). The maximisation of the log-likelihood is typically a hard task since it is a multidimensional surface possibly containing many local maxima. For this reason, it is important to run the optimisation algorithm several times from different initial values to make sure a global maximum is found.

3.3.3. Gaussian process surrogate

Once the correlation lengths have been estimated, their value is assumed known and substituted into Equation (3.45). This gives the final form of the posterior for the simulator as:

$$\eta(\mathbf{x}) \,|\, \mathbf{y} \sim t_{n-q}\left(M^{**}(\mathbf{x}; \hat{\boldsymbol{\delta}}), \hat{\sigma}^2 C^{**}(\mathbf{x}, \mathbf{x}'; \hat{\boldsymbol{\delta}})\right). \tag{3.52}$$

The GP methodology outlined here has become known as the 'plug-in' approach due to the treatment of the correlation lengths. This method does not fully account for the uncertainty in these hyperparameters and code uncertainty will be underestimated as a consequence. However, Kennedy and O'Hagan (2001) argue that only a second-order effect of uncertainty about these hyperparameters is neglected, and the large computational savings of the plug-in approach outweigh these disadvantages. Furthermore, studies by Bayarri et al. (2007) show that the plug-in approach tends to give very similar results to the fully Bayesian alternative, for a reduced computational cost.

The posterior in Equation (3.52) is a fully probabilistic surrogate for the simulator. It is known as a GP emulator and will be denoted $\hat{\eta}^{GP}(\cdot)$ in this work. The posterior mean function, $M^{**}(\mathbf{x}) \equiv M^{**}(\mathbf{x}; \hat{\boldsymbol{\delta}})$, interpolates the simulator runs $\mathbf{y}$, subject to a small error from the nugget term. It can be evaluated cheaply at any new input setting $\mathbf{x}^{(*)}$ to provide a prediction of the simulator output $\eta(\mathbf{x}^{(*)})$.

The posterior covariance function, denoted as $V^{**}(\mathbf{x}, \mathbf{x}') = \hat{\sigma}^2 C^{**}(\mathbf{x}, \mathbf{x}'; \hat{\boldsymbol{\delta}})$, provides a measure of uncertainty where the simulator has not yet been run, quantifying code uncertainty. The uncertainty is effectively zero at the experimental design points, subject to a small amount of uncertainty due to the inclusion of a nugget term. This reflects the deterministic nature of the simulator. In general terms, the posterior uncertainty becomes larger the further away from the observed simulator runs.

As with PC, the GP emulator enables fast Monte Carlo (or more efficient quadrature method) estimation of any function, $g(\cdot)$, of the uncertain simulator output, Y, as:

$$\mathbb{E}\left[g(Y)\right] \approx \frac{1}{m}\sum_{i=1}^{m} g(\hat{\eta}^{GP}(\mathbf{x}^{(i)})), \tag{3.53}$$

for a suitably large sample of size m from the input distribution $f_{\mathbf{X}}(\mathbf{x})$. This MC estimation is most simply performed by using the posterior mean function $M^{**}(\mathbf{x})$ in place of the full probabilistic surrogate $\hat{\eta}^{GP}(\mathbf{x})$[2]. However, because the GP surrogate is a full probability distribution, any summary of the simulator output is itself a random variable. The posterior covariance function $V^{**}(\mathbf{x}, \mathbf{x}')$ can be employed, or the full GP emulator can be sampled from, to form uncertainty statements for the estimation in Equation (3.53), such as a confidence interval. A method for doing this will be presented in Section 4.2 in Chapter 4.

3.3.4. Examples

In this section, GP surrogates are applied to simple one-dimensional and two-dimensional examples. In particular, GP emulators are built for exactly the same examples as given in Section 3.2.4 for PC surrogates, for comparison of the two methods. Recall the one-dimensional simulator:

$$y = \eta(x) = 5 + 5x + \sin(5 + 5x),$$

as already given in Equation (3.25), and consider building GP surrogates with squared exponential and Matérn correlation functions. For simplicity, the mean function is fixed at $M(x, \boldsymbol{\beta}) = 0$ (equivalent to choosing no regression basis functions, $q = 0$) for all examples. This leaves only two hyperparameters to estimate in deriving the GP posteriors: the GP variance σ^2, and the correlation length δ in the correlation functions. A small nugget parameter of size 1×10^{-7} is also added to the diagonal elements of the correlation matrix to aid numerical conditioning when building each GP emulator. The size of the nugget was chosen arbitrarily to be small enough not to visibly affect the interpolating property of the GP emulator, whilst providing adequate numerical conditioning.

In the equivalent example for PC surrogates, design sizes of $n = 2, 3, 4, 5$ were used to fit PCEs with truncation orders $p = 1, 2, 3, 4$. Furthermore, two design types were used for building PC surrogates using the two non-intrusive methods for estimating the coefficients: Sobol sequence designs for the regression technique, and Gauss-Legendre quadrature rules for the non-intrusive spectral projection approach. In this example, exactly the same design types and sizes are used to fit the GP emulators, for comparison to the PC examples in Section 3.2.4.

Figure 3.8 shows the posteriors for squared exponential and Matérn GP surrogates, for the Sobol sequence designs. For all design sizes and types, the posterior mean function of the GP surrogates interpolates the simulator runs with effectively zero

[2] Although doing so would introduce bias if Equation (3.53) is nonlinear.

uncertainty, subject to a small error from the nugget parameter (not visible here). Posterior uncertainty, which is represented as two posterior standard deviations around the posterior mean, $M^{**}(x) \pm 2\sqrt{V^{**}(x,x)}$, quantifies code uncertainty. That is, it gives an estimate of the uncertainty where the simulator has not yet been run, due to the fact that the simulator has only been observed at n points. In particular, this uncertainty is large when the GP surrogate extrapolates outside the region of the observed simulator runs. As more design points are added, the posterior mean function becomes closer to the true function, and uncertainty is reduced in between the observed data, for both choices of correlation function. The posterior mean function is similar for the squared exponential and Matérn cases, although the uncertainty is generally larger across the input domain for the Matérn correlation function. This is because the Matérn correlation function assumes a twice differentiable function rather than an infinitely differentiable one (as is the case for the squared exponential correlation function), so more variation is expected. The function from Equation (3.25) is in fact infinitely differentiable, so the squared exponential correlation function is the optimal choice here, but this would not be known in practice.

Figure 3.9 shows the posteriors for squared exponential and Matérn GP surrogates, for the Gauss-Legendre quadrature rule designs. In general, the same behaviour for the GP surrogates can be observed as more design points are added. The principle difference here is that the Gauss-Legendre quadrature rule design places points regularly and symmetrically across the input domain, as opposed to the Sobol sequence designs which add design points sequentially and at different distances from each other. The Gauss-Legendre design points also explore nearer to the extremities of the input domain, meaning that uncertainty is lower in these regions. A potential problem of the use of evenly spaced design points is demonstrated by the GP emulators built on the design of size $n = 4$, where the design points have fallen in such a way that the GP emulators have very smooth posterior mean functions with low uncertainty. In other words, the emulators are over-confident about their poor predictions. However, this is just an unfortunate aspect of that particular design, and PC surrogates also fit a similar surrogate (see Figure 3.4).

Secondly, consider building GP surrogates for the two-dimensional simulator:

$$y = \eta(x_1, x_2) = \exp(-x_1)\tanh(5x_2),$$

as already given in Equation (3.27). As with the one-dimensional example, GP emulators with squared exponential and Matérn correlation functions are considered, and the mean function is fixed at $M(\mathbf{x}; \boldsymbol{\beta}) = \beta_1 = 0$. The remaining hyperparameters of the GP are estimated using simulator runs: the GP variance σ^2, and the two correlation lengths, δ_1 and δ_2, in the correlation functions. A small nugget parame-

Figure 3.8.: Gaussian process posterior mean functions (blue) for the one-dimensional example $y = \eta(x) = 5 + 5x + \sin(5 + 5x)$ (black), using Sobol sequence designs. For all cases, the prior mean function is a constant $\beta_1 = 0$. Posterior uncertainty is represented as two posterior standard deviations around the posterior mean, $M^{**}(x) \pm 2\sqrt{V^{**}(x, x)}$, and plotted across the input domain $x \in [-1, 1]$ (grey shading). The rows correspond to design sizes of $n = 2, 3, 4, 5$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.

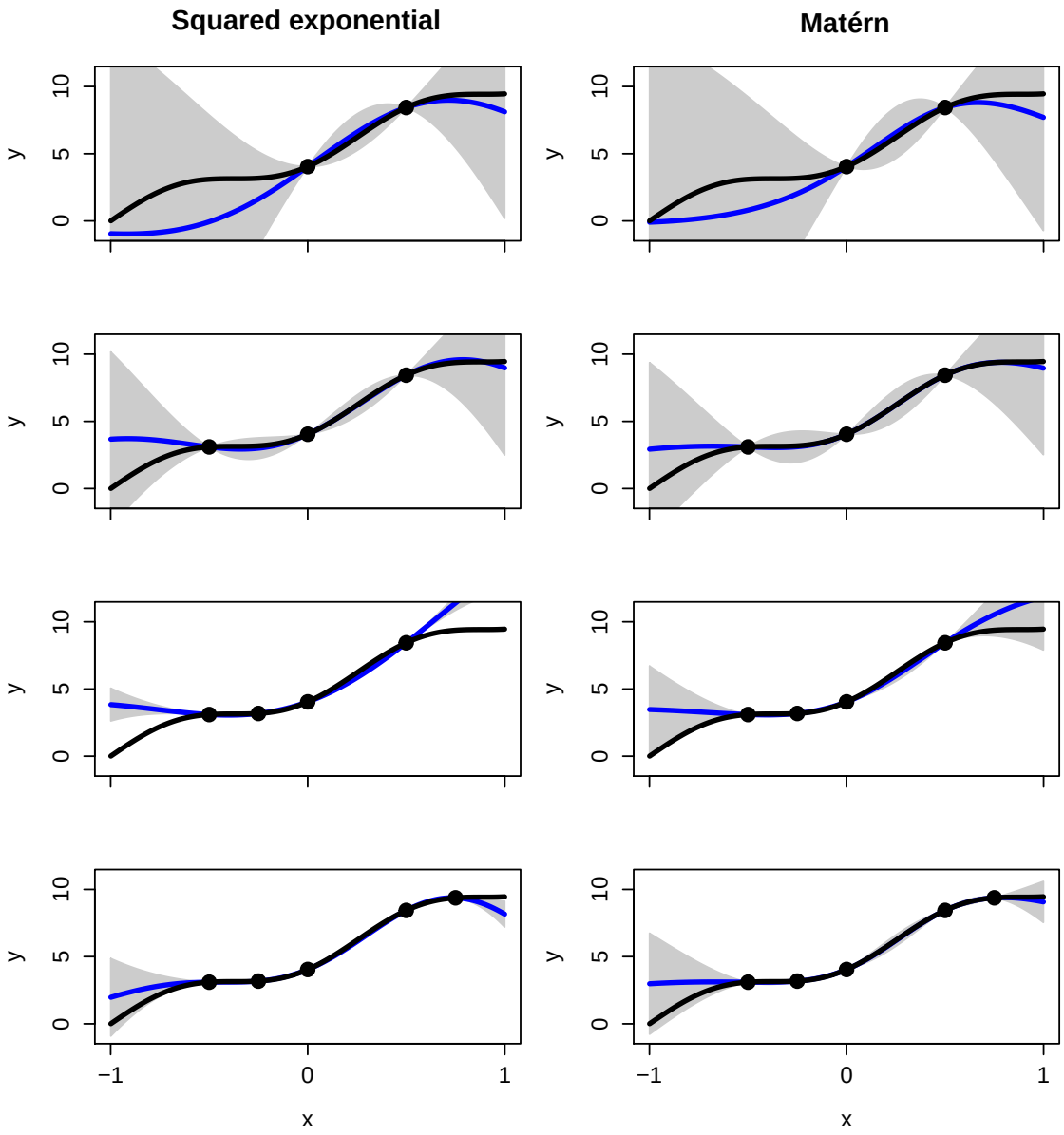
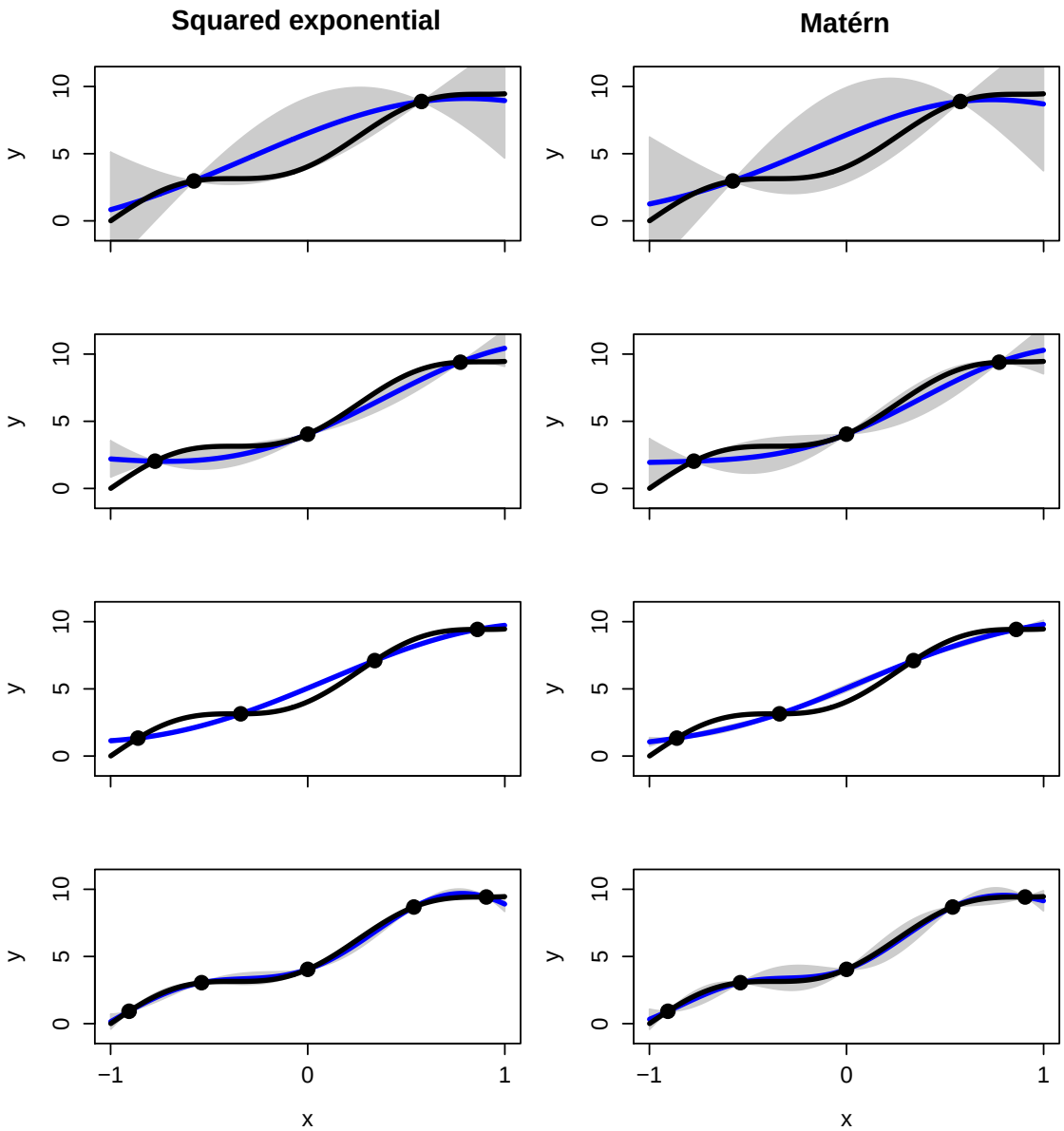


Figure 3.9.: Gaussian process posterior mean functions (blue) for the one-dimensional example $y = \eta(x) = 5 + 5x + \sin(5 + 5x)$ (black), using Gauss-Legendre quadrature rule designs. Posterior uncertainty is represented as two posterior standard deviations around the posterior mean, $M^{**}(x) \pm 2\sqrt{V^{**}(x, x)}$, and plotted across the input domain $x \in [-1, 1]$ (grey shading). The rows correspond design sizes of $n = 2, 3, 4, 5$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.



ter of size 1×10^{-7} is again added to the diagonal elements of the correlation matrix to aid numerical conditioning when building each GP emulator.

In the equivalent example for PC surrogates, design sizes of $n = 4, 9, 16, 25$ were used to fit PCEs with truncation orders $p = 1, 2, 3, 4$. Sobol sequence and Gauss-Legendre quadrature rules were again used as design types to build the PC surrogates, to align with the regression and non-intrusive spectral projection methods for estimating the PC coefficients respectively. In particular, a tensor grid design of one-dimensional Gauss-Legendre quadrature rules was used for the non-intrusive spectral projection case. In this example, exactly the same design types and sizes are used to fit the GP emulators, for comparison to the PC examples in Section 3.2.4.

Figures 3.10 and 3.11 show the posterior mean functions and posterior uncertainty respectively, for the GP surrogates built on Sobol sequence designs. The uncertainty is represented by the twice the posterior standard deviation, $2\sqrt{V^{**}(\mathbf{x}, \mathbf{x})}$, at each point $\mathbf{x} = (x_1, x_2)$ in the two-dimensional grid. Similar to the PC experiment, as the design size increases the posterior mean function of the GP surrogate becomes closer to the true function. The posterior uncertainty clearly decreases across the grid as design points are added, giving a measure of confidence in the surrogate prediction. The difference in performance for the two correlation functions is small in this case.

Figures 3.12 and 3.13 show equivalent plots for GP surrogates built on tensor grid designs. As for the PC experiment, the structure of the tensor grid design gives a symmetrical appearance of the resulting GP surrogate mean functions. Again, as more design points are added, the posterior mean becomes closer to the true function. However, the regularity of the tensor grid results in substandard behaviour of the posterior standard deviation, which does not necessarily decrease as more design points are added. Furthermore, clear bands of equal standard deviation can be observed along the main directions of the grid. In particular, the GP emulators built on the design of size $n = 9$ have very smooth posterior mean functions and low uncertainty. Like the example of $n = 4$ for the one-dimensional example, this is another unfortunate aspect of that particular design and not due to the poor performance of either the PC or GP methods.

3.4. Discussion

Polynomial chaos and Gaussian process emulation both enable efficient surrogate-based uncertainty quantification for expensive simulators. Originating and used in different UQ communities, they are contrasting approaches to essentially the same problem. As seen in Section 3.2, PC is an applied mathematics and engineering ap-

Figure 3.10.: Gaussian process posterior mean functions (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image), using Sobol sequence designs. For all cases, the prior mean function is a constant $\beta_1 = 0$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.

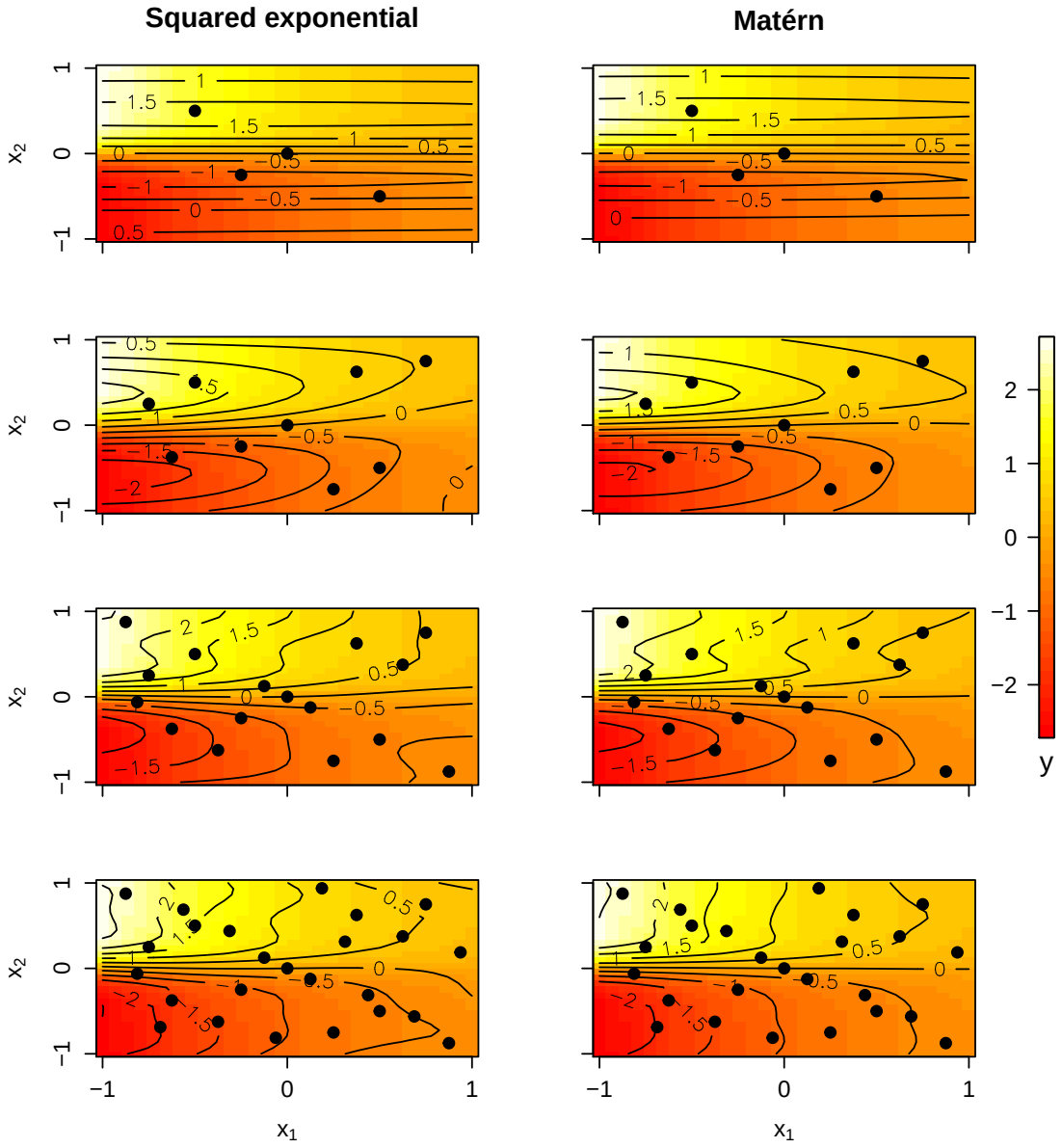


Figure 3.11.: Gaussian process posterior uncertainty (coloured image) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (not shown), using Sobol sequence designs. Posterior uncertainty is represented as two posterior standard deviations (2 SD), $2\sqrt{V^{**}(\mathbf{x}, \mathbf{x})}$, and plotted across the input domain $\mathbf{x} = (x_1, x_2) \in [-1, 1]^2$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.

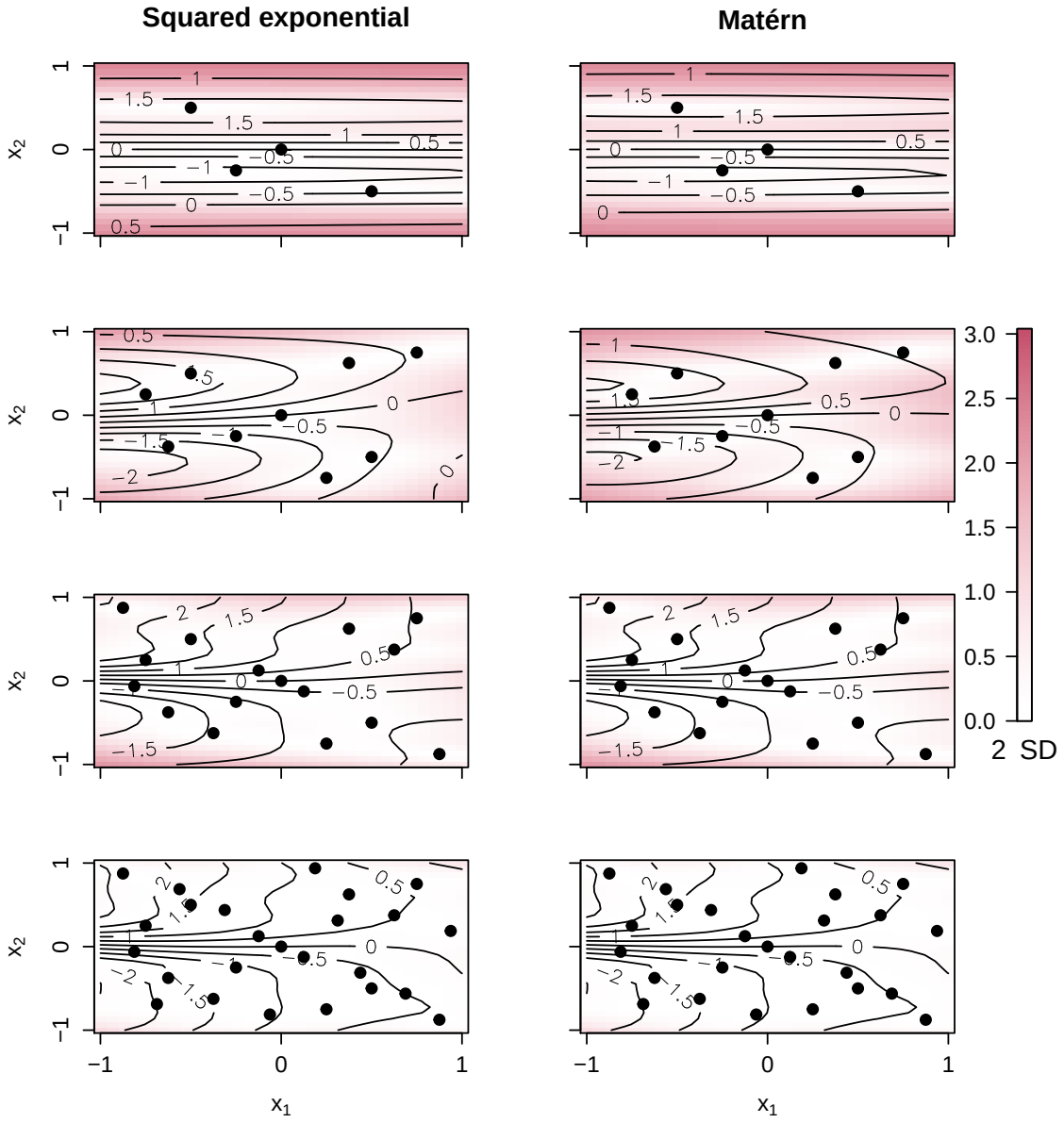


Figure 3.12.: Gaussian process posterior mean functions (black contour lines) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (coloured image), using tensor grid designs of Gauss-Legendre quadrature rules. For all cases, the prior mean function is a constant $\beta_1 = 0$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.

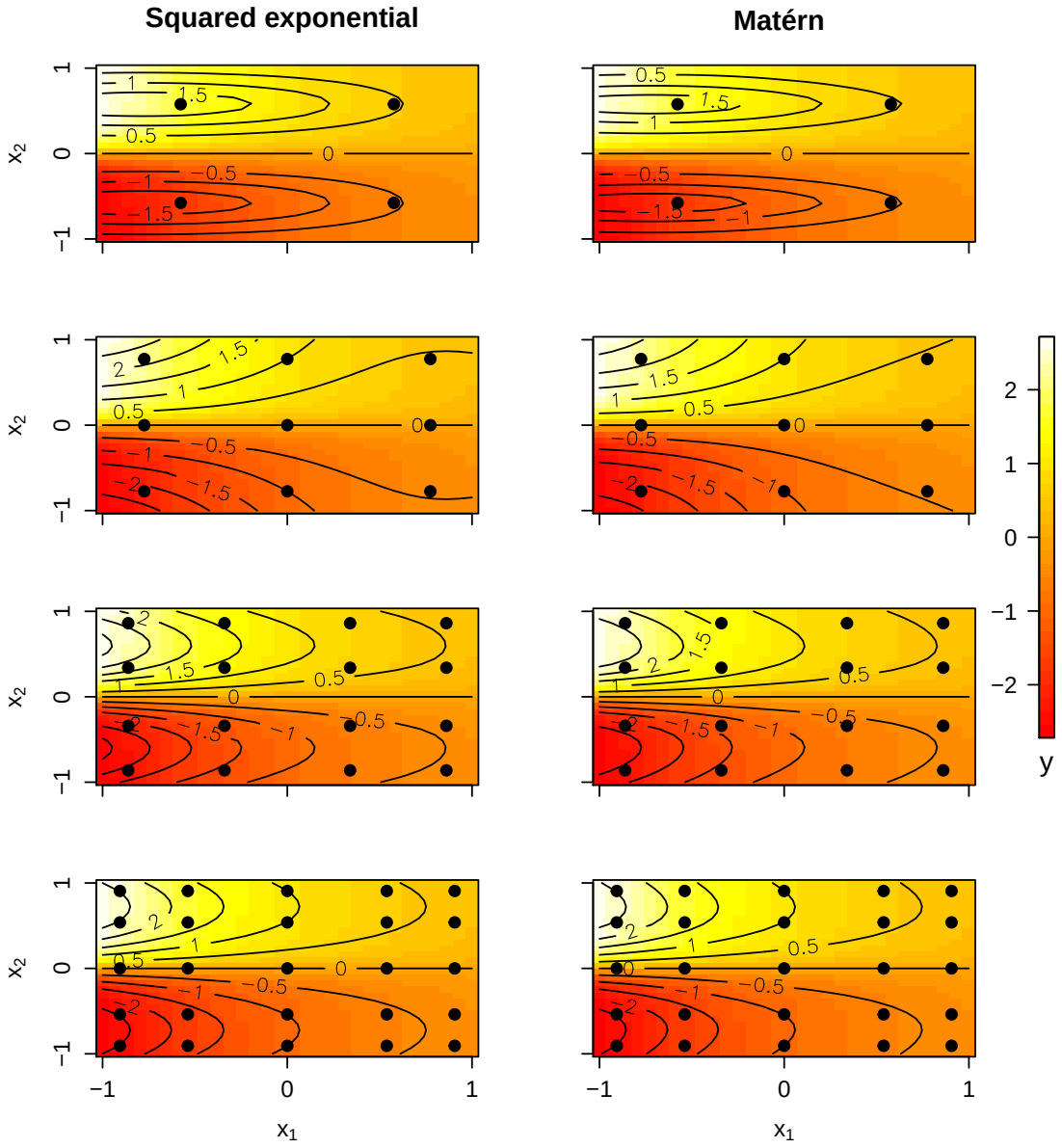
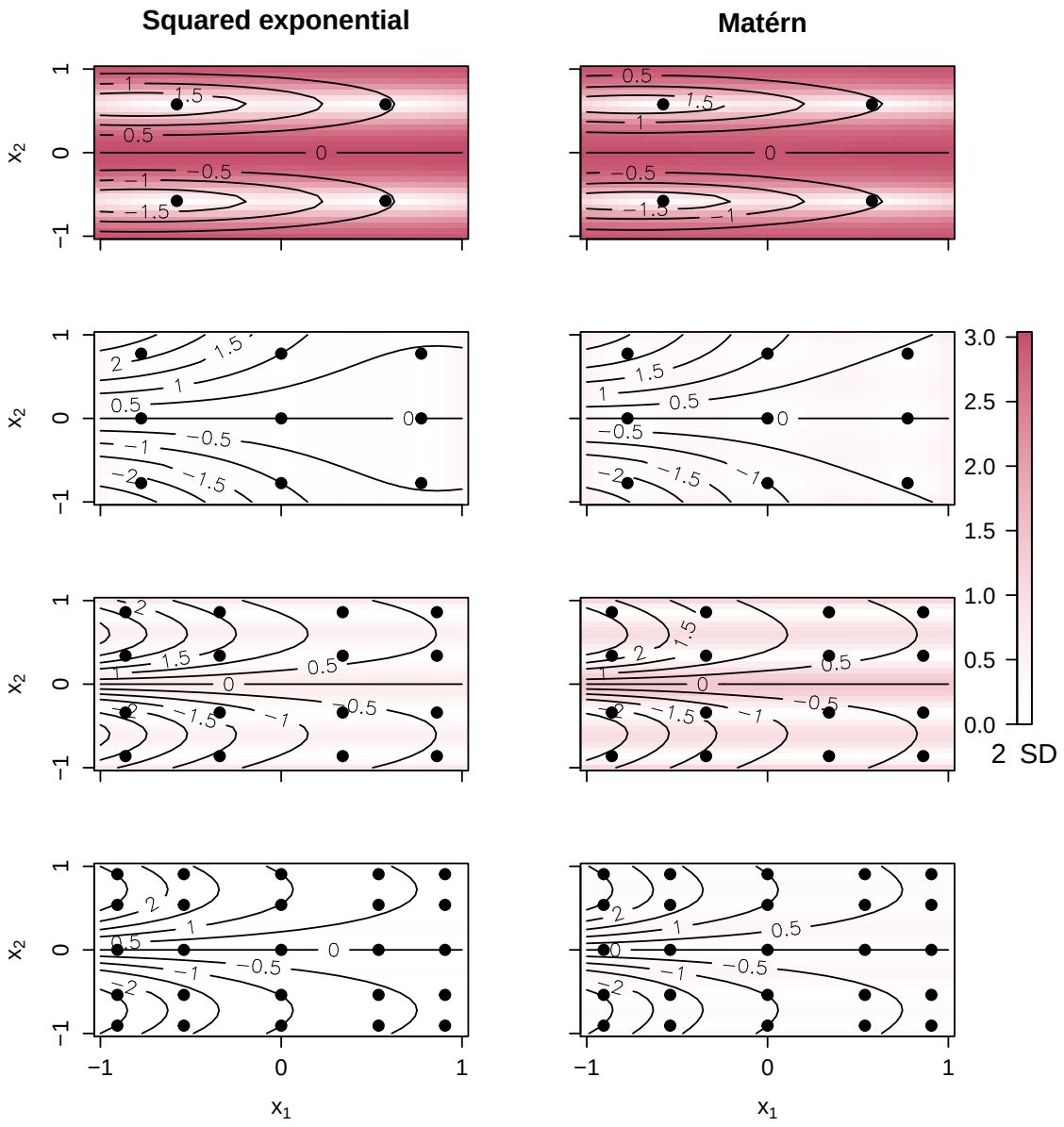


Figure 3.13.: Gaussian process posterior uncertainty (coloured image) for the two-dimensional example $y = \eta(\mathbf{x}) = \exp(-x_1)\tanh(5x_2)$ (not shown), using tensor grid designs of Gauss-Legendre quadrature rules. Posterior uncertainty is represented as two posterior standard deviations (2 SD), $2\sqrt{V^{**}(\mathbf{x}, \mathbf{x})}$, and plotted across the input domain $\mathbf{x} = (x_1, x_2) \in [-1, 1]^2$. The rows correspond to design sizes of $n = 4, 9, 16, 25$, and the columns refer to squared exponential and Matérn choices for the correlation function. The experimental design is shown as black dots in each case.



proach which builds a polynomial surrogate using orthogonal polynomial families. On the other hand, Section 3.3 showed that GP emulation is a Bayesian statistical approach, which models the simulator using a prior distribution and updates it into a posterior distribution using simulator runs. Examples shown in Section 3.2.4 and 3.3.4 for PC and GP surrogates respectively highlighted that the surrogate methods perform quite differently even for simple one-dimensional and two-dimensional examples.

Despite having such different appearances, it was remarked by Schöbi et al. (2015) that PC (specifically the regression case) is actually a special case of GP emulation. This can be observed by setting the GP prior mean function to be a truncated PCE:

$$M(\mathbf{x};\mathbf{a}) = \sum_{0\leq|\boldsymbol{\alpha}|\leq p} a_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\mathbf{x}), \tag{3.54}$$

and setting the prior correlation function to be the Dirac delta function:

$$C(\mathbf{x},\mathbf{x}') = \delta(\mathbf{x}-\mathbf{x}'). \tag{3.55}$$

Equivalently, this is achieved by letting the correlation length hyperparameters $\boldsymbol{\delta} \to \mathbf{0}$. In this case, the GP reduces down to a simple regression model of polynomial terms as the correlation matrix $\mathbf{A} = \mathbf{I}_n$.

Recalling the four properties a surrogate should possess outlined in Section 3.1 — fast, deterministic, uncertainty and quality — allows a general assessment of PC and GP emulation as surrogate models. Firstly, both PC and GP emulation provide fast approximations for the simulator, and can quickly predict the simulator output at an untried input configuration. Secondly, PC surrogates only satisfy the deterministic property when $n = N$ (for both the regression and NISP cases), that is, when the design size is equal to the number of coefficients in the PCE. When $n > N$, the interpolating property is relaxed. In this work, it is always the case that $n = N$ for the PC surrogates built using NISP, but the regression alternatives are often built on designs of size $n > N$ so will not be interpolating functions. GP surrogates interpolate the design points, subject to a small error from the nugget parameter, regardless of design size (provided that the design size is greater than or equal to the number of hyperparameters to estimate). Next, PC surrogates do not provide any uncertainty information since they are just a polynomial function. On the other hand, GP surrogates are a full posterior distribution for the simulator so uncertainty statements can be made. In particular, uncertainty where the simulator has not yet been run (code uncertainty) is quantified. Moreover, the uncertainty is effectively zero at the experimental design points (again subject to the nugget term) to reflect the deterministic quality of the simulator.

Since GP emulators satisfy the first three of the conditions, whereas PC surrogates satisfy only one, it could be concluded that GP emulators are the better surrogate approach. However, this is before considering the final condition outlined in Section 3.1, and perhaps the most important property of a surrogate model: its quality. That is, how accurate is the surrogate at approximating the simulator? At a glance, the PC and GP surrogates built for the one-dimensional and two-dimensional examples in Sections 3.2.4 and 3.3.4 both provide quality approximations to the simulators, even for small design sizes. There is no doubt though, that the two surrogates provided different approximations to one another, that is, they were accurate in different parts of the input domain, even when built on identical designs. There is a need for a more in depth comparison of the two methods in terms of their quality.

The quality of a surrogate can be measured in a number of ways. The important stage of surrogate model validation — testing whether a surrogate is a good approximation to the simulator — is the subject of Chapter 4. In particular, strategies for validation are different within the PC and GP communities, and this has inhibited a critical comparison of the two approaches. In the following chapter, the approximation accuracy of PC and GP is compared for two simulators used in industry, using an unbiased set of validation metrics. At the time of writing, this work is believed to be the first critical and direct comparison of the two methods in the UQ literature. Such a comparison is intended to provide useful information for a practitioner in UQ as to which method may be more accurate in different scenarios.

Finally, there are a number of other important issues associated with the use of surrogate models, including:

- What are the computational costs, and are there any numerical issues, when building the surrogate?
- How flexible is the surrogate in representing a range of simulator behaviours?
- Is it easy to rebuild the surrogate if assumptions about the simulator change?
- How easy is it to use the surrogate in practice? Does it need to be adapted to tackle objectives in UQ?

These questions are also addressed independently in the PC and GP literatures, and there is a lack of research comparing the two approaches. While the comparisons in Chapter 4 are designed mainly to test the approximation accuracy of the two surrogate approaches, it is also hoped that some light can also be shed towards the comparative abilities of the surrogates concerning these points.

4. Comparison of surrogate methods

In this chapter, the accuracy of polynomial chaos and Gaussian process emulators in approximating simulator output is compared in a range of modelling scenarios. The chapter is structured as follows. Section 4.1 outlines a range of ways that surrogate models can be validated, and a short literature review of specific validation techniques for PC and GP methods is given in Sections 4.1.1 and 4.1.2 respectively. Validation metrics used in PC and GP communities are shown to be contrasting and prohibit comparison of the two methods. To tackle this issue, a set of simple validation metrics applicable to both PC and GP techniques are outlined in Section 4.2. A suite of experiments are then designed and described in Section 4.3 which make use of the validation metrics. The experiments aim to investigate how the comparative performance of the two approaches changes with variations in the size and type of the experimental design used to build the surrogates. Results from these experiments are given in Section 4.4 for two simulators used in industry, named `adJULES` and `VEGACONTROL`, along with some discussion. The chapter is concluded by giving some general advantages and disadvantages of the two approaches in Section 4.5, before a conclusion in Section 4.6. Most of the results in this chapter are published in Owen et al. (2017).

4.1. Surrogate model validation

Once a polynomial chaos surrogate, $\hat{\eta}^{PC}(\cdot)$, or a Gaussian process emulator, $\hat{\eta}^{GP}(\cdot)$, has been built, it is important to assess its quality. That is, how well does the surrogate approximate the simulator? Are the assumptions made when building the surrogate satisfied? How accurate are the surrogate's predictions of the simulator output at untried input settings? Answering these questions is the process of surrogate model validation. It is vital to validate the surrogate, otherwise any inference drawn from subsequent surrogate-based UQ analysis for the simulator could be invalid.

Surrogate model validation falls into two categories. Firstly, simulator output at an

independent validation design can be used to test the quality of the surrogate. A validation design is a set of m additional input settings for the simulator, denoted $\mathcal{D}' = \left(\mathbf{x}'^{(1)}, \ldots, \mathbf{x}'^{(m)}\right)$. The corresponding simulator output at the validation design is $\mathbf{y}' = \left(y'^{(1)}, \ldots, y'^{(m)}\right)^\top$, obtained by evaluating $y'^{(i)} = \eta(\mathbf{x}'^{(i)})$ for $i = 1, \ldots, m$. The simulator output at the validation design can be compared to corresponding surrogate predictions at these points, $\hat{\eta}(\mathbf{x}'^{(i)})$, $i = 1, \ldots, m$, using various metrics. It is important that the validation design points are distinct from the original experimental design $\mathcal{D}$, whilst remaining draws from the probability distribution of the inputs ($\mathbf{x}'^{(i)} \sim f_{\mathbf{X}}(\mathbf{x})$, $i = 1, \ldots, m$), to fully test the quality of the surrogate. The main advantages of this first validation approach are that the surrogate model does not have to be rebuilt or modified in any way, and surrogate predictions at the validation design can be obtained cheaply (Bastos and O'Hagan, 2009). However, this strategy requires m additional runs of the simulator, which may not be possible for expensive simulators. Furthermore, m must be large enough (100,1000,...) to obtain good validation accuracy. Nonetheless, it is common to include room for a validation design in the computational budget of a UQ project.

The second validation strategy involves reusing the original experimental design points in some way. The main class of methods in this case are cross-validation techniques. Here, a subset of points from the experimental design are withheld when building the surrogate. The surrogate is then used to predict the output at the withheld input settings, which can be compared to the simulator output using appropriate metrics. The process is repeated for all permutations of withheld points of the experimental design, and an average of the validation metrics is taken. The most common example here is leave-one-out (LOO) cross-validation, where a single point of the experiment design is withheld each time (Sammut and Webb, 2011). The main advantage of this approach is that no extra simulator runs are required, but computational cost does arise from refitting the surrogate model for each permutation of the experimental design. Another potential disadvantage of this approach is that an optimal design of n points, which is used to build the surrogate initially, may be very suboptimal for $n - 1$ points.

In practice, a combination of the above two strategies can be used to assess the quality of the surrogate. In either strategy, a key choice is the validation metric used to compare surrogate predictions to the corresponding simulator output. There are many metrics available for this purpose, which may test the ability of the surrogate in estimating summary statistics of the simulator output, or how well the surrogate approximates the simulator across the input space. With regards to PC and GP surrogate methodologies, it is not surprising that the validation metrics used are different in each case, since they have been developed independently in separate UQ communities. To highlight this, a short review of the validation techniques used in

the PC and GP fields is given in the following sections.

4.1.1. Polynomial chaos

The PC surrogate, given by the truncated PCE in Equation (3.22), converges in the mean-square sense as $p \to \infty$ (Xiu and Karniadakis, 2003). However, to be used in practice the truncation order p must be selected by the practitioner according to the computational budget and any beliefs about the complexity of the simulator. Common practice is to fix p at finite order and experiment with several values (for example, $p = 1, 2, 3, 4$) until some kind of convergence is observed. Ideally, the truncation error induced by truncating the PC expansion at order p should be small:

$$Y - \hat{\eta}^{PC}(\mathbf{X}) = \sum_{|\boldsymbol{\alpha}| \geq p+1} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}). \tag{4.1}$$

However, it is not possible to calculate the truncation error. Instead, a number of general diagnostics can be used to assess convergence of the PCE (Sudret, 2008; O'Hagan, 2013):

- Monitor the difference between equivalent coefficient values as p is increased. For example, in the one-dimensional setting the coefficient associated with the polynomial $\psi_1(x)$, a_1, will converge as $p \to \infty$[1]. Setting a tolerance threshold for the change in coefficients as p increases can highlight when convergence has been achieved.

- Check the coefficients associated with the p^{th}-degree polynomial terms. If all coefficients are below a small threshold then a lower order expansion is satisfactory.

- Check estimates of the statistical moments of the simulator obtained from the coefficients, such as the expectation and variance given by Equations (3.23) and (3.24) respectively. Setting a tolerance threshold for the change in these estimates as p increases can indicate convergence.

Aside from these general diagnostics concerned with the PCE itself, there exist a number of validation metrics comparing the PC surrogate to the simulator. A general metric used in the surrogate modelling community is the generalisation error (Schöbi et al., 2015), defined as the expectation of the squared difference between the simulator and the surrogate:

$$Err_{gen} = \mathbb{E}\left[(Y - \hat{\eta}(\mathbf{X}))^2\right], \tag{4.2}$$

[1] Although since $n \geq p$, this will not be possible!

where the expectation is taken with respect to the distribution $f_{\mathbf{X}}(\mathbf{x})$. Clearly, Err_{gen} close to zero indicates good agreement between simulator and surrogate. An analytical solution to Equation (4.2) is not possible, since the simulator cannot be evaluated everywhere. However, if an additional set of simulator runs is available in a validation design $\mathcal{D}'$, the generalisation error can be estimated as follows:

$$\widehat{Err}_{gen}^{PC} = \frac{1}{m}\sum_{i=1}^{m}\left(y'^{(i)} - \hat{\eta}^{PC}(\mathbf{x}'^{(i)})\right)^2. \tag{4.3}$$

It is important that the validation design, $\mathcal{D}'$, is distinct from the experimental design used to build the surrogate, $\mathcal{D}$, to fully test the quality of the surrogate.

If it is not possible to obtain additional runs of the simulator in a validation experiment, the generalisation error can also be estimated using the original experimental design points in a cross-validation strategy. For example, the generalisation error could be estimated using the LOO cross-validation error, defined as:

$$Err_{LOO}^{PC} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{\eta}_{-i}^{PC}(\mathbf{x}^{(i)})\right)^2, \tag{4.4}$$

where $\hat{\eta}_{-i}^{PC}(\cdot)$ is a PC surrogate built on the experimental design with the point $\mathbf{x}^{(i)}$ withheld. Note that the LOO error is not applicable for a PC surrogate built using NISP, since all experimental design points (quadrature points) must be used for the method to work.

Classical diagnostics from linear regression also apply directly to the PC surrogate built using regression. These include the R^2 or adjusted R^2 coefficients of determination (Faraway, 2006) which give the percentage of variance in simulator output explained by the surrogate, as well the F-statistic to test the proposed surrogate against the null model or alternative surrogates. Graphical diagnostics for the residuals of the PC regression may also prove useful.

4.1.2. Gaussian process emulation

The fact that a GP emulator is a probabilistic surrogate introduces some interesting questions with regards to its validation. Namely, should just the posterior mean function be compared to the simulator, or should the posterior uncertainty also be incorporated? If the posterior mean function is the main focus, then the validation metrics introduced for PC in Section 4.1.1 easily cross over to the GP surrogate. Recalling that the posterior mean function is denoted as $M^{**}(\mathbf{x})$, an estimate of the

generalisation error in Equation (4.2) using the validation design is:

$$\widehat{Err}_{gen}^{GP} = \frac{1}{m}\sum_{i=1}^{m}\left(y'^{(i)} - M^{**}(\mathbf{x}'^{(i)})\right)^2. \tag{4.5}$$

to be compared with the PC equivalent in Equation (4.3). Again, the validation design must be distinct from the design used to build the GP emulator to fully test the surrogate.

If it is not possible to obtain additional runs of the simulator in a validation design, a cross-validation strategy can be used. Similarly to version given for PC in Equation (4.4), the generalisation error could be estimated using the following LOO cross-validation error:

$$Err_{LOO}^{GP} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - M_{-i}^{**}(\mathbf{x}^{(i)})\right)^2, \tag{4.6}$$

where $M_{-i}^{**}(\cdot)$ is the posterior mean function of the GP emulator built on the experimental design with the point $\mathbf{x}^{(i)}$ withheld.

Due to the distributional assumptions made when constructing a GP emulator, any summary statistic of the emulator (such as the metrics given in Equations (4.5) and (4.6)), is itself a random variable. Hence, uncertainty information such as credible intervals for the summary statistic can be derived. A numerical procedure for doing this was presented by Oakley and O'Hagan (2002), and will be presented in Section 4.2.1.

A set of validation diagnostics specifically for GP emulators were given by Bastos and O'Hagan (2009), who argued that traditional metrics — such as the estimated generalisation error in Equation (4.5) — are not appropriate since emulator predictions are not independent. In fact, emulator predictions are correlated, with the correlation known exactly through the posterior covariance function. The diagnostics they proposed take this correlation into account, and are particularly tailored to investigating whether the assumptions made in building a GP emulator are satisfied. These include the assumptions of normality and stationary, as well as the restrictions on global behaviour and differentiability of the function implied through the choices of mean and covariance functions. Even when these assumptions are reasonable, hyperparameters of the mean and covariance functions may be estimated poorly due to a bad choice of experimental design. The correlation length hyperparameters are also fixed at the final stage of building the GP emulator, rather than their uncertainty taken into account, which may not be appropriate. The diagnostics given by Bastos and O'Hagan (2009) allow the user to see whether these assumptions hold and whether hyperparameters have been estimated properly. They rely on access to a validation design, rather than a cross-validation approach. Many of the diagnostics are based around the so-called standardised prediction errors (SPEs), which

are defined as:

$$SPE_i = \frac{y'^{(i)} - M^{**}(\mathbf{x}'^{(i)})}{\sqrt{V^{**}(\mathbf{x}'^{(i)}, \mathbf{x}'^{(i)})}}, \qquad i = 1, \ldots, m, \tag{4.7}$$

that is, the difference between the simulator output and the posterior mean, standardised by the posterior variance, at the validation design points. Note that the SPEs are a generalisation of the $\widehat{Err}^{GP}_{gen}$ metric given in Equation (4.5). Under the distributional assumptions made, the SPEs have standard Student's-t distributions conditional on the simulator runs and plug-in estimates of the correlation lengths. Large errors (absolute value larger than 2) can highlight nonstationarity, poor choice of the mean function and estimation of the associated hyperparameters, as well as overestimation or underestimation of the covariance function hyperparameters. Since the SPEs are correlated with one another, Bastos and O'Hagan (2009) also suggest several variance decompositions of the errors, including the eigen, Cholesky, and pivoted Cholesky decompositions. Plotting the decomposed SPEs against the emulator's predictions, validation data index or each simulator input can be particularly useful graphical diagnostics for identifying problems with the GP surrogate. If a single summary diagnostic is required, Bastos and O'Hagan (2009) suggest the use of the Mahalanobis distance, which is defined as:

$$M_{dist} = (\mathbf{y}' - M^{**}(\mathcal{D}'))^{\top} (V^{**}(\mathcal{D}', \mathcal{D}'))^{-1} (\mathbf{y}' - M^{**}(\mathcal{D}')), \tag{4.8}$$

where $M^{**}(\mathcal{D}')$ is a vector of the posterior means, and $V^{**}(\mathcal{D}', \mathcal{D}')$ a matrix of posterior covariances, at the validation design $\mathcal{D}'$. Under GP emulator assumptions, the distribution of M_{dist}, conditional on the simulator runs and plug-in estimates of the correlation lengths, is a scaled F-Snedecor distribution with m and $n-q$ degrees of freedom (Bastos and O'Hagan, 2009):

$$\frac{n-q}{m(n-q-2)} M_{dist} \,|\, \mathbf{y}, \boldsymbol{\delta} \sim F_{m,n-q}. \tag{4.9}$$

Large or small values of the Mahalanobis distance can indicate a conflict between the GP emulator and the simulator. If this is the case, it is important to investigate the SPEs to locate the cause of the problem.

4.2. Methods for comparing surrogates

The validation diagnostics for PC and GP emulation, presented in Sections 4.1.1 and 4.1.2 respectively, both study the residuals of the surrogates' predictions of the simulator output in some way. Clearly, the accuracy of the surrogates can be directly compared using the appropriate estimates of the generalisation error Err_{gen}, either at a validation design (Equations (4.3) and (4.5)) or through LOO

cross-validation (Equation (4.4) and (4.6)). For both the independent validation design and cross-validation cases, it is critical that the same designs are used for building and validating PC and GP surrogates for a fair comparison. However, it is clear that the PC and GP communities also have their own specific validation strategies, whether it be the general diagnostics of convergence of the PC surrogate given in Section 4.1.1, or the Bastos and O'Hagan (2009) metrics for GP emulation given in Section 4.1.2. This has perhaps discouraged a comparison of PC and GP in the UQ literature to date, and poses a problem for one of the primary objectives of this work: the comparison of the contrasting methodologies in different modelling scenarios. It is important that the PC and GP surrogates are compared in a fair and unbiased manner, without favouring either approach through the use of particular validation metrics. For example, the GP diagnostics given by Bastos and O'Hagan (2009) which take correlated predictions into account are not appropriate for the PC surrogate.

As remarked at the end of Chapter 3, the main focus of the comparison in this work is in the ability of the two approaches at approximating the simulator — the prediction UQ objective outlined in Section 2.2.1. In particular, it is desired to compare GP emulators (with different choices for the mean and covariance functions) to the two PC surrogate approaches outlined in Section 3.2 — regression and NISP. To do this, in this work a validation strategy based on an independent validation design is used. Therefore, in addition to the experimental design $\mathcal{D}$ used to build a surrogate, a further set of m simulator runs in a validation design $\mathcal{D}'$ is required. This approach is favoured over the cross-validation strategy for a number of reasons. Firstly, for the simulators featured in the examples in the remainder of this chapter, it is relatively cheap to produce a set of validation runs. Secondly, as mentioned previously the cross-validation approach is not compatible with the NISP method for PC, since all points in the quadrature rule must be used. Related to this issue is the fact that the surrogate methods may be built on different design types (for example, a Sobol sequence for PC regression and a Gaussian quadrature rule for PC NISP), and it is not clear what effect this would have on cross-validation diagnostics where different design points would have to be withheld in each case. When using a validation design, the validation metric for each surrogate is calculated at exactly the same set of points — regardless of how the surrogate has been constructed — so is considered to be a fairer comparison. In all cases, the validation design $\mathcal{D}'$ is chosen to be a Latin Hypercube design in d inputs with size $m = 1000$.

With the validation strategy decided, it remains to choose a set of validation metrics. The accuracy of the surrogate in approximating the simulator can be tested in various ways. To measure how close the surrogate is to the simulator across the

input space, the root mean square error (RMSE) is used:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y'^{(i)} - \hat{\eta}(\mathbf{x}'^{(i)})\right)^2}. \tag{4.10}$$

The RMSE is simply the square root of an estimate of the generalisation error given in Section 4.1, often used due to the fact that it has the same units as the simulator output. Since this in effect measures the distance between the surrogate and simulator over the input space, a lower RMSE is preferable. Common practice is to standardise the RMSE by the variance of the simulator output at the simulator design, but this is not done here for simplicity.

It is also important for the surrogate to be able to estimate summary statistics of the simulator output for the uncertainty analysis UQ objective. Let μ_Y and σ_Y denote the true values of the simulator output mean and standard deviation respectively. These can be estimated empirically from the surrogate as follows:

$$\mu_Y \approx \hat{\mu}_Y = \frac{1}{m}\sum_{i=1}^{m}\hat{\eta}(\mathbf{x}'^{(i)}), \tag{4.11}$$

$$\sigma_Y \approx \hat{\sigma}_Y = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\hat{\eta}(\mathbf{x}'^{(i)}) - \hat{\mu}_Y\right)^2}, \tag{4.12}$$

Note that these quantities can also be estimated directly from the PC coefficients using Equations (3.23) and (3.24). However, the above expressions are used for both PC and GP surrogates for fairer comparison of the methods, since the same information is used to estimate the mean and variance in each case.

To test the capability of the surrogates in estimating the probability of rare events, the exceedance probabilities $Pr(Y \geq \mu_Y + \kappa\sigma_Y)$ are also estimated. Specifically, $\kappa = 2, 3$ for probabilities of exceeding two and three standard deviations above the mean respectively. These quantities can be estimated from the surrogate as follows:

$$Pr(Y \geq \mu_Y + \kappa\sigma_Y) \approx \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}\left(\hat{\eta}(\mathbf{x}'^{(i)}) \geq \mu_Y + \kappa\sigma_Y\right), \qquad \kappa = 2, 3, \tag{4.13}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function. This quantity can be obtained directly for GP emulation by integrating the posterior, but once again the above expression is used for both PC and GP surrogates for fairer comparison of the approaches.

Where the RMSE is preferred to be as small as possible, the summary statistics in Equations (4.11), (4.12) and (4.13) must be compared to corresponding simulator values. However, the values of μ_Y, σ_Y and $Pr(Y \geq \mu_Y + \kappa\sigma_Y)$ are not known exactly and must also be estimated at the validation design $\mathcal{D}'$. To account for uncertainty

on simulator quantities induced by the use of finite ($m = 1000$) evaluations, a bootstrap analysis is performed to obtain 95% bootstrap percentile intervals (Efron and Tibshirani, 1994).

Finally, the ability of the surrogate in estimating the PDF of the simulator output is assessed. This is done by smoothing surrogate predictions at the validation design points using a kernel density estimator. For all examples, a Gaussian kernel is used and the kernel bandwidth is estimated from the simulator output at the validation design using the method outlined in Silverman (1986). The kernel bandwidth estimated from the simulator output itself is then kept fixed when smoothing the PC and GP predictions at the validation design for a consistent comparison.

When calculating all of the above validation metrics, for PC $\hat{\eta}(\mathbf{x}^{(i)}) = \hat{\eta}^{PC}(\mathbf{x}^{(i)})$ as given by Equation (3.22), and for GP $\hat{\eta}(\mathbf{x}^{(i)}) = M^{**}(\mathbf{x}^{(i)})$, that is, the posterior mean function from Equation (3.52).

4.2.1. Oakley and O'Hagan (2002) method

Due to the fact that a GP emulator is a probabilistic surrogate for the simulator, any summary statistic of the emulator is itself a random variable with a probability distribution. Samples from this probability distribution can be obtained numerically using the method given in Oakley and O'Hagan (2002). Their method works by repeatedly drawing samples from the GP emulator at a simulation design, $\mathcal{S} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(s)}\right)$. The algorithm is as follows:

For k in $1 : K$,

1. Generate pseudodata $\mathbf{y}_{(k)} = \left(\eta_{(k)}(\mathbf{x}^{(1)}), \ldots, \eta_{(k)}(\mathbf{x}^{(s)})\right)$ by simulating from the GP emulator (given by the posterior distribution in Equation (3.52)) at the simulation design $\mathcal{S}$.

2. Obtain the posterior distribution $\eta_{(k)}(\mathbf{x}) \,|\, \mathbf{y}, \mathbf{y}_{(k)}$, by building a GP emulator using data $(\mathcal{D}, \mathcal{S})$ and $\left(\mathbf{y}, \mathbf{y}_{(k)}\right)$, assuming no error on the pseudodata.

3. Approximate $\eta_{(k)}(\mathbf{x})$ with its posterior mean function from Step 2, $M^{**}_{(k)}(\mathbf{x})$.

4. Calculate the desired validation metric using $M^{**}_{(k)}(\mathbf{x})$ at the validation design $\mathcal{D}'$.

The resulting K estimates from Step 4 are a sample from the distribution of the validation metric. Obtaining the posterior from Step 2 is cheap since the mean and covariance function hyperparameters have already been estimated, so can be treated as known. Furthermore, if the simulation design is chosen appropriately the variance of $\eta_{(k)}(\mathbf{x})$ is small, meaning that the approximation in Step 3 is minimal.

In the experiments to follow, the Oakley and O'Hagan (2002) algorithm is employed for GP emulators to generate samples from the distributions of the RMSE, mean, standard deviation, exceedance probabilities and PDF validation metrics given in Section 4.2. Specifically, $K = 1000$ and a 95% confidence interval is constructed by taking the 2.5% and 97.5% percentiles of the resulting sample from the probability distribution of each validation metric. The simulation design $\mathcal{S}$, used for Step 1 of the Oakley and O'Hagan (2002) algorithm, is a Latin Hypercube in d inputs with size $s = 100$.

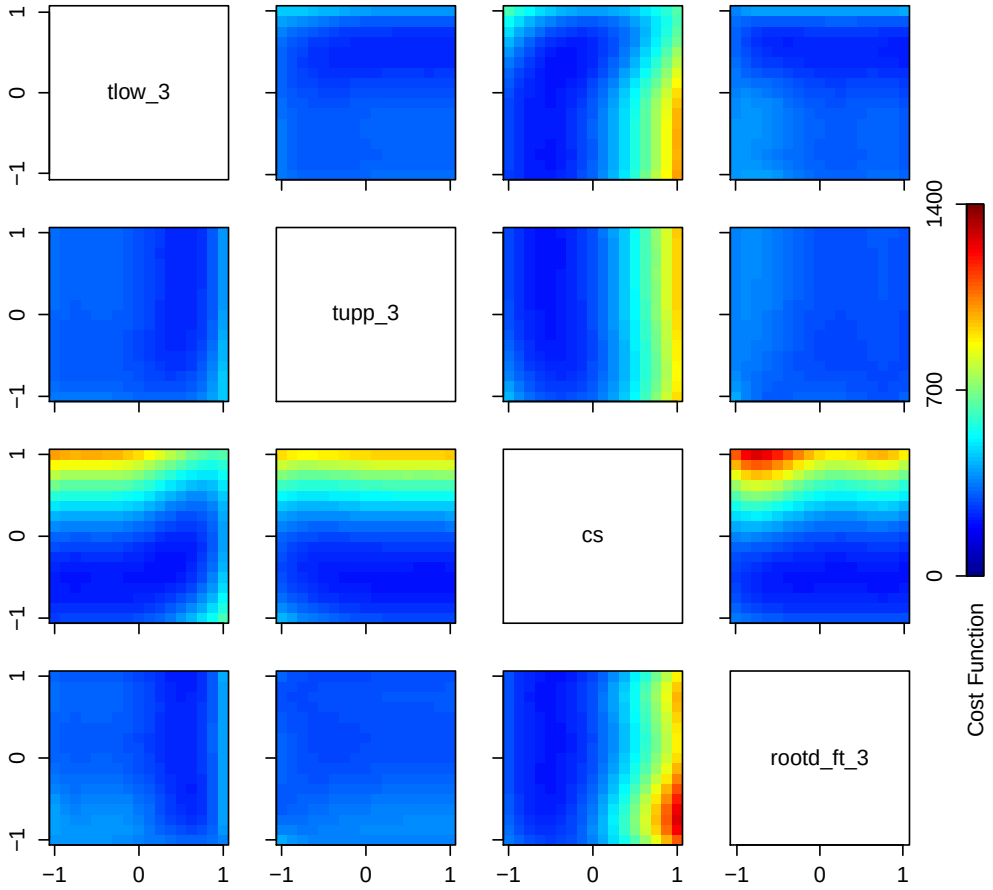
4.3. Experiments

With the validation strategy chosen, it remains to describe the experiments in which PC and GP surrogates are compared. In this work, the approximation accuracy of the surrogates are compared for two simulators used in industry, named `adJULES` and `VEGACONTROL`. A description of the two simulators is given in Sections 4.3.1 and 4.3.2 respectively. Following this, the experimental designs for each simulator are given in Section 4.3.3, as well as more detail about the experiments.

4.3.1. adJULES simulator

The Joint UK Land Environment Simulator (`JULES`) (Best et al., 2011; Clark et al., 2011) is a simulator which models the interactions between the land and the atmosphere. It is the land surface component currently used in the UK Met Office Unified Model. `JULES` uses a number of meteorological drivers and vegetation processes (for example, photosynthesis and soil microbial activity) to model radiation, heat, water and carbon fluxes. Presently, observed time series of these fluxes cannot be incorporated into the `JULES` framework. A new system, `adJULES` (Raoult et al., 2016), has been developed to provide this functionality and also comprises an adjoint model for parameter estimation and optimisation studies. The current implementation of `adJULES` contains nine inputs detailing various plant properties and a single output, a cost function for optimisation purposes. Expert elicitation prior to the experiment led to the focus on $d = 4$ of the most important inputs: `t_low`, `t_upp`, `cs` and `rootd_ft`. These represent lower and upper temperatures for photosynthesis (°C), carbon content of soil ($\mathrm{kg\,C\,m^{-2}}$) and root depth (m) respectively. The inputs are transformed to the range $[-1, 1]$, so that $\mathcal{X} \in [-1, 1]^4$, and assumed to be uniformly distributed on this domain. The remaining five inputs are kept at their nominal values when running the `adJULES` simulator, which is treated as a black-box. The aim is to build surrogate models for the output (cost function) as a function of the four inputs. A visualisation of the `adJULES` simulator output

Figure 4.1.: Visualisation of the adJULES simulator output (cost function) as a function of pairwise combinations of each of the four inputs. The smoothing of the simulator output is performed using the surrogate with the lowest root mean square error in the adJULES experiments (see Figure 4.3), which is the the polynomial chaos expansion built on the largest class 3 design (see Table 4.1).

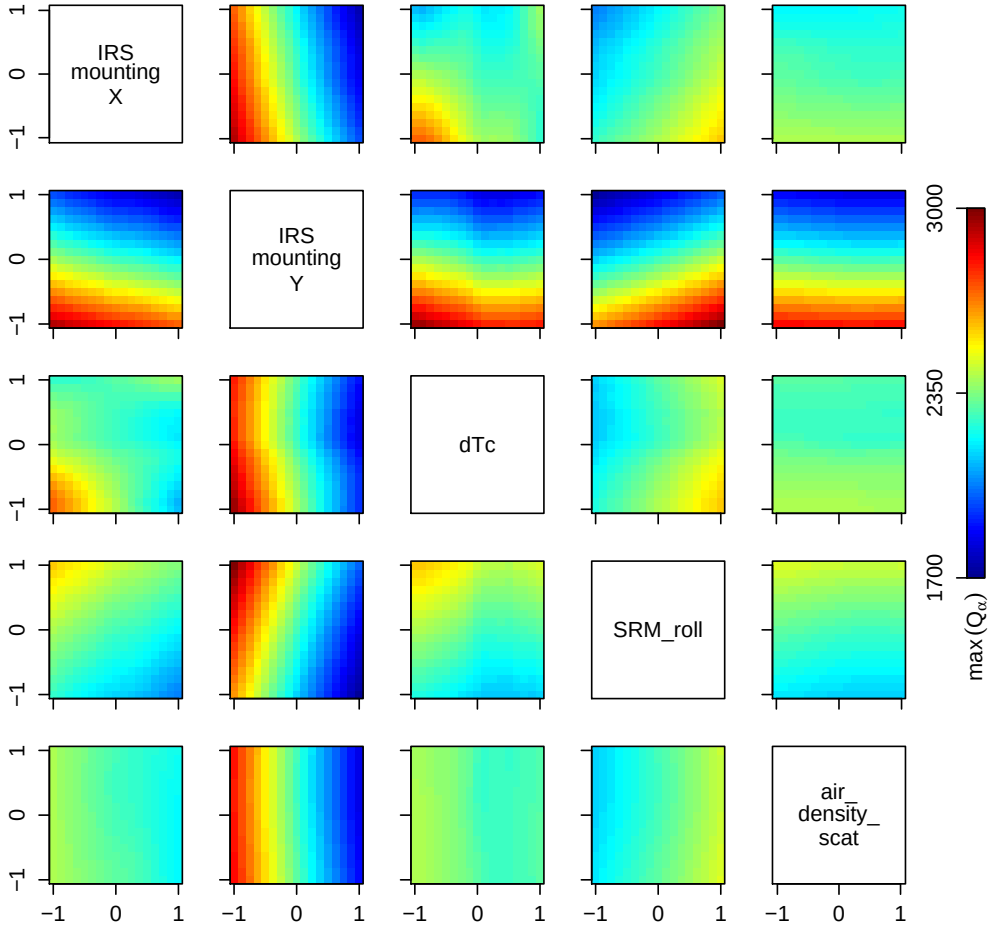


as a function of pairwise combinations of each of the four inputs is shown in Figure 4.1.

4.3.2. VEGACONTROL simulator

`VEGACONTROL` is a simulator used by the European Space Agency to model the VEGA launch vehicle (rocket) in its atmospheric and exo-atmospheric flight phases. The flight phase with altitude between 30m and 60km is considered in the present study. The equations for motion include force and drag components depending on Mach and angle of attack, kinematic coupling in all axes, and a nonlinear model of the electromechanical actuator dynamics with associated backlash and delays. The simulator comprises several modes which describe different processes affecting flight, including: launch vehicle stability, control of thrust and roll, atmosphere dynamics, amongst many others. The current implementation of `VEGACONTROL` has 83 inputs and 37 outputs. Expert elicitation prior to the experiment selected a single important simulator output, namely the maximum of the aerodynamic load over the entire

Figure 4.2.: Visualisation of the VEGACONTROL simulator output ($\max(Q_\alpha)$) as a function of pairwise combinations of each of the five inputs. The smoothing of the simulator output is performed using the surrogate with the lowest root mean square error in the VEGACONTROL experiments (see Figure 4.7), which is the Gaussian process emulator with Matérn correlation function built on the largest class 2 design (see Table 4.1).



flight phase. This is denoted $\max(Q_\alpha)$, where Q is the dynamic pressure and α is the angle of attack. Furthermore, $d = 5$ influential inputs were chosen: `IRSmountingX`, `IRSmountingY`, `dTc`, `SRM_roll` and `air_density_scat`. These correspond to the Indian Remote Sensing (IRS) mounting error with respect to the X and Y body axes, scattering on time burn, scattering on roll degree, and atmospheric density respectively. The inputs are transformed to the range $[-1, 1]$, so that $\mathcal{X} \in [-1, 1]^5$, and assumed to be uniformly distributed on this domain. The remaining inputs are kept at their nominal values when running the `VEGACONTROL` simulator, which is treated as a black-box. The objective is to build surrogate models for the output ($\max(Q_\alpha)$) as a function of the five inputs. A visualisation of the `VEGACONTROL` simulator output as a function of pairwise combinations of each of the five inputs is shown in Figure 4.2.

Table 4.1.: Summary of the designs used in the adJULES and VEGACONTROL experiments. The design size n increases to allow polynomial chaos expansions with larger truncation orders p. A description of the design classes is given in Section 4.3.3. Note that a fourth class 3 design was not implemented for the VEGACONTROL experiment due to computational restrictions. Sobol sequence designs of size 625 and 1024 were also used for the adJULES and VEGACONTROL experiments respectively for comparison to the largest tensor grid designs.

Class	Type	PC type	p	n_{adJULES}	$n_{\text{VEGACONTROL}}$
Class 1	Sobol sequence	Regression	1	5	6
			2	15	21
			3	35	56
			4	70	126
Class 2	Sobol sequence	Regression	1	10	12
			2	30	42
			3	70	112
			4	140	252
Class 3	Tensor grid	NISP	1	16	32
			2	81	243
			3	256	1024
			4	625	—

4.3.3. Experimental set-up

The purpose of these experiments is to investigate how the comparative approximation accuracy of PC and GP surrogates, measured by the validation metrics in Section 4.2, varies with changes in the experimental design. In particular, the performance of the two surrogate approaches is tested under changes in the type and size of the experimental design. The rationale behind the choices of design will now be given. A summary of the designs used in the `adJULES` and `VEGACONTROL` experiments is presented in Table 4.1. Firstly, the experimental design types have been grouped into three classes to distinguish between different PC approaches. Recall in Section 3.2.2 that two strategies were outlined for estimating the coefficients of the PCE: regression and non-intrusive spectral projection (NISP). For the regression approach, there are no restrictions on the experimental design, but one should generally use a space-filling design with enough design points to ensure a well-determined linear system (the solution to which is given in Equation (3.18)). For this purpose, Sobol sequence designs (Niederreiter, 1988) are used, and these make up class 1 and class 2 designs (the difference between the two will be explained shortly). Sobol sequences are used — instead of the more popular Latin Hypercube designs — because of the fact that taking the first n points of the full design preserves the Sobol sequence property. In this way, computational time can be saved by evaluating the simulator at one large Sobol sequence design, and using the first n points to build surrogates on smaller design sizes. For the NISP approach, Gaussian quadrature rules are used to estimate the coefficients. The experimental designs in this case are tensor (factorial) grid designs made up of one-dimensional Gauss-Legendre quadrature rules (since all

inputs are assumed to be uniformly distributed) in each of the d dimensions, and these make up class 3 designs.

Secondly, each design class comprises four distinct designs of increasing size. Similar to the class of design, the design sizes are chosen according to how the PC surrogate is constructed. Recall in Section 3.2.1 that the design size (n) for a PC surrogate must be at least as big as the number of terms in the PCE (N), which itself depends on the truncation order (p), the input dimension (d), and the truncation scheme used. In summary, $n \geq N$, where $N = \binom{d+p}{p}$ for total order truncation (used in the regression case) and $N = (p+1)^d$ for tensor product truncation (used in the NISP case). Since the input dimension has been fixed for the `adJULES` ($d = 4$) and `VEGACONTROL` ($d = 5$) simulators, the truncation order is varied in the following experiments. In particular, PC surrogates are built with $p = 1, 2, 3, 4$ ($p > 4$ is not considered due to computational restraints) for the four designs in each class. This fixes the number of terms in each of the PC surrogates, and all that remains is to choose suitable design sizes. For the tensor product designs in class 3, naturally $n = N$, but for Sobol sequence designs in class 1 and class 2 any value of $n \geq N$ is permitted. In the following experiments, $n = N$ for class 1 designs and $n = 2N$ for class 2 designs. Class 1 and class 2 designs are referred to as uniquely-determined and twice over-determined respectively, due to the relationship between the design size and the number of terms in the PC surrogate. Hosder et al. (2007) advocate for the rule of thumb that $n = 2N$, so class 1 designs test PC methods when faced with small design sizes. To give an example from Table 4.1, for the `adJULES` experiment $d = 4$, therefore a third-order PC surrogate ($p = 3$) uses the following design sizes in each class: $n = N = \binom{4+3}{3} = 35$ (class 1); $n = 2N = 2 \times \binom{4+3}{3} = 70$ (class 2); $n = N = (3+1)^4 = 256$ (class 3).

For each design in Table 4.1, polynomial chaos surrogates are built using the appropriate method (regression or NISP) to estimate the coefficients. From this point on, all polynomial chaos surrogates built using the different non-intrusive techniques are referred to by the acronym PC for ease of comparison to GP methods.

Gaussian process emulators are also built for each design size and type in Table 4.1. As is common in the literature, the mean function for each GP emulator is chosen using a stepwise regression algorithm which selects the most important basis functions from constant, linear and quadratric trends in each of the inputs, as well as first order interactions between them[2] (Rougier et al., 2009a; Vernon et al., 2010). The number of basis functions, q, is naturally restricted by the design size n and the additional $d + 1$ hyperparameters to estimate (the d correlation lengths $\boldsymbol{\delta}$ and the

[2]This is similar to the sparse polynomial chaos methodology of Blatman and Sudret (2010a), which selects important terms in the polynomial chaos expansion. This is not used in the regression approach for polynomial chaos in these experiments for simplicity.

GP variance σ^2). Specifically, $q \leq n - (d + 1)$, and this is respected in each case. For each design in Table 4.1, two separate GP emulators are built with different correlation functions. In particular, the squared exponential correlation function in Equation (3.35) is used since it is the standard choice in the literature. In case the assumption of infinite differentiability does not hold, the Matérn correlation function in Equation (3.36) is also used, which assumes a twice-differentiable function. In the following experiments, these separate GP emulators are referred to by the acronyms SE GP and M GP respectively.

Since the sizes of the largest class 3 designs in each experiment are much bigger than any of the class 1 or 2 designs, Sobol sequences of size 625 and 1024 were also tested in the `adJULES` and `VEGACONTROL` experiments respectively. This is to test whether there is any difference in the choice of experimental design when a larger number of runs can be afforded. Note also that a rule of thumb in the GP literature is to use a design size ten times that of the input dimension ($n = 10d$) (Loeppky et al., 2009). Since $d = 4$ and 5 for the `adJULES` and `VEGACONTROL` simulators respectively, many of the design sizes featured here are smaller than this suggestion. Hence, the accuracy of GP surrogates is also tested when faced with small design sizes.

To recap the set-up of the experiments, PC, SE GP and M GP surrogates are built on all the designs in Table 4.1, as well as the large Sobol sequence designs mentioned in the previous paragraph. For both the `adJULES` and `VEGACONTROL` experiments, an independent Latin Hypercube design of size $m = 1000$ is also generated for validation purposes. Simulator output at the validation design points are compared with predictions from each of the surrogate methods. For each surrogate method and design, the RMSE, mean, standard deviation, exceedance probability and PDF validation metrics are computed as described in Section 4.2. Crucially, the surrogates are built and compared on exactly the same designs for a fair comparison. Surrogate estimates of the mean, standard deviation, exceedance probabilities and PDF are compared to corresponding simulator quantities evaluated from the validation design (with bootstrap error), whereas the RMSE is preferred to be as low as possible.

4.4. Results

4.4.1. adJULES experiment

Validation results from fitting PC, SE GP and M GP surrogates to the `adJULES` simulator are presented in Figures 4.3, 4.4, 4.5 and 4.6. Firstly, consider results for the RMSE, mean and standard deviation validation metrics (given by Equations (4.10)–(4.12)) shown in Figure 4.3. It is clear for class 1 designs that both GP em-

ulators outperform PC surrogates regardless of the validation metric. The accuracy of PC surrogates does not necessarily improve with design size (and consequently, the addition of more polynomial terms), suggesting that the uniquely-determined regression approach is unstable. In particular, attention is drawn to the PC surrogate built on the third design of this class (size 35). For this design class, there is little difference between the SE GP and M GP methods, and they are the preferred surrogates. For class 2 designs, the use of twice over-determined regression to fit the PC surrogates leads to more interesting results. When analysing results for the RMSE metric, there is little to choose between PC and GP approaches. It can be noted that PC has a faster initial reduction in error, but this does not continue to improve. Both GP emulators have a slower reduction in error, but may begin to outperform PC methods for larger design sizes than featured here. Concerning the simulator mean, all surrogate estimates are within the calculated confidence intervals, and uncertainty on GP emulator estimates can be seen to reduce as the design size increases. As such, none of the surrogate models would be preferred over one another. For the simulator standard deviation, PC and GP estimates are mostly similar, although it can be noted that PC estimates tend to be closer to the 'true' value. However, uncertainty on the GP estimates themselves does mean that it is difficult to choose a favoured surrogate approach here. For class 3 designs, it can be observed that PC methods are consistently more accurate than the GP approaches with regards to the RMSE metric, so they are preferred here. When estimating the simulator mean, results are very close to those for class 2 designs, in that all surrogates produce similar estimates that are within the calculated confidence intervals. Some preference for PC methods can be found in the case of standard deviation metric, as GP estimates are unstable to increases in design size whereas PC estimates gradually become more accurate. This may be due to poor estimation of the variance and correlation length hyperparameters of the GP, as is often the case for tensor grid designs (Urban and Fricker, 2010).

Secondly, consider the exceedance probability validation metrics (given by Equation (4.13)) presented in Figure 4.4. For class 1 designs, again it is clear that GP methods outperform PC surrogates for all design sizes and both exceedance probabilities. However, note that while GP estimates are comparatively better and preferred in this case, they are still not particularly accurate. Furthermore, the uncertainty on GP estimates is very small, meaning that they are over confident about their inaccurate predictions. Clearly, for these small design sizes it is difficult to get an accurate estimate of such small probabilities; no surrogate can get within the calculated confidence intervals for the probability of exceeding three standard deviations above the mean. For class 2 designs, there is a large improvement for PC in estimating both exceedance probabilities, and they arguably give more accurate estimates since they fall within the calculated confidence intervals more often than GP approaches. More

Figure 4.3.: Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the adJULES simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines).

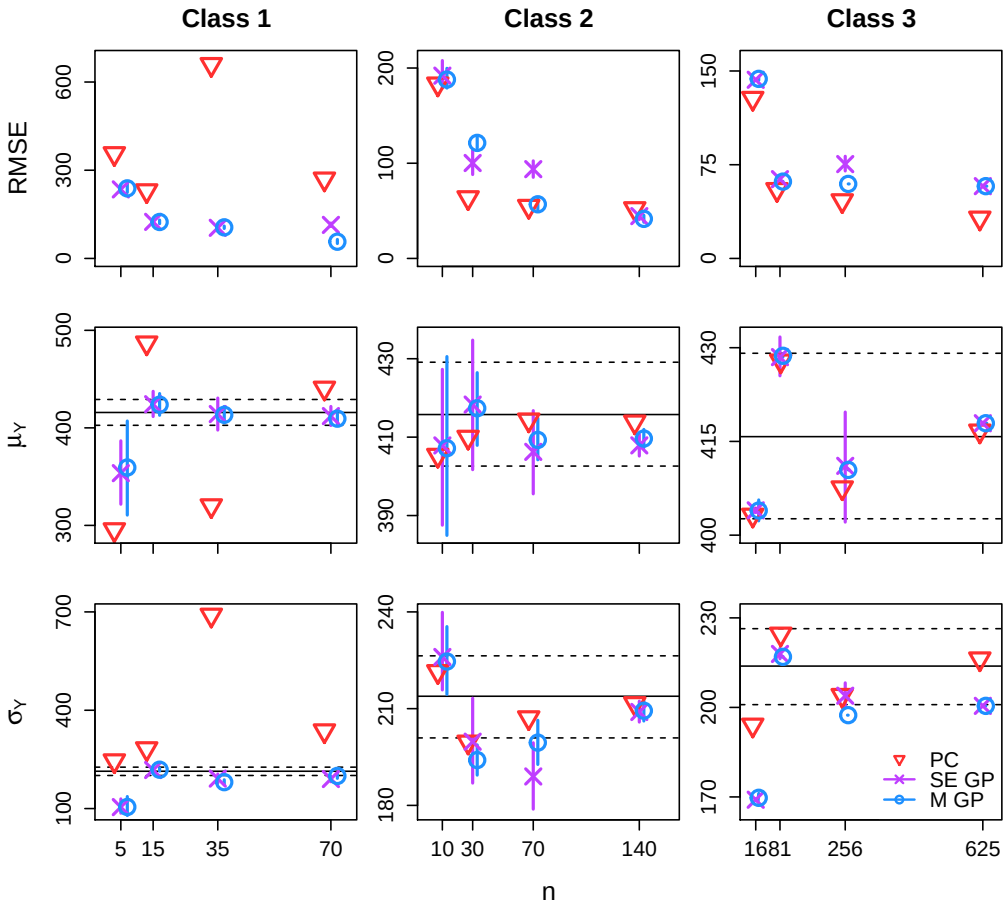
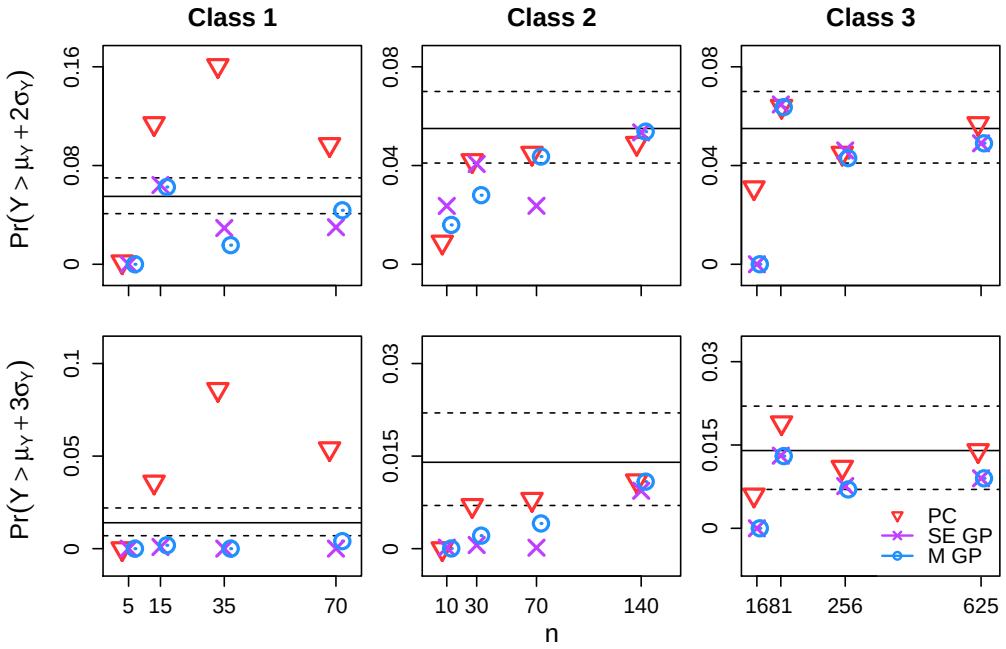


Figure 4.4.: Exceedance probabilities $Pr(Y > \mu_Y + 2\sigma_Y)$ and $Pr(Y > \mu_Y + 3\sigma_Y)$ as a function of design size n and design class for the adJULES simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines).



stability can also be observed here compared to class 1, with accuracy generally improving as design size is increased for all surrogates, with the exception of the SE GP method. For class 3 designs, results are similar for PC and GP methods, with their estimates falling within the confidence intervals apart from for the smallest design in the class. It could perhaps be said here that PC methods do better at estimating the probability of exceeding three standard deviations above the simulator mean.

Now consider the PDF validation metrics shown in Figure 4.5. As observed with the previous validation metrics, a clear preference for GP methods can be found in the case of class 1 designs, where PC methods struggle to estimate the PDF accurately for all design sizes. For class 2 and 3 designs, it is difficult to find any differences between the methods, although a slight preference for PC can be seen for individual designs (for example, the second and third designs in class 2 and the first design in class 3). For large design sizes in these classes, all surrogates estimate the PDF to a high degree of accuracy.

Finally, PC, SE GP and M GP surrogates were also built on a Sobol sequence design of size 625 for the `adJULES` simulator, to see if any improvement could be made on the validation results from the largest tensor grid design. RMSE, mean and standard deviation metrics for surrogates built on tensor grid and Sobol sequence designs of size 625 are shown in Figure 4.6 (top panels). It is observed that GP

Figure 4.5.: Probability density function (PDF) validation metrics for the adJULES simulator. Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates.

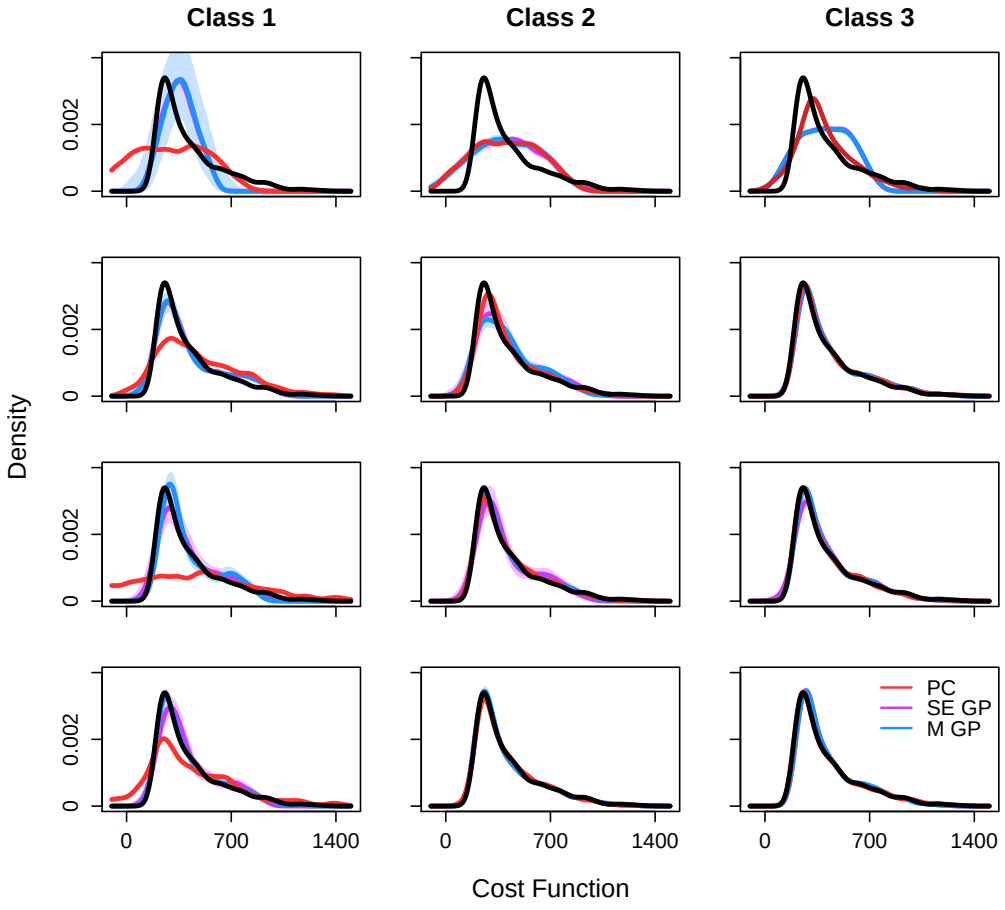
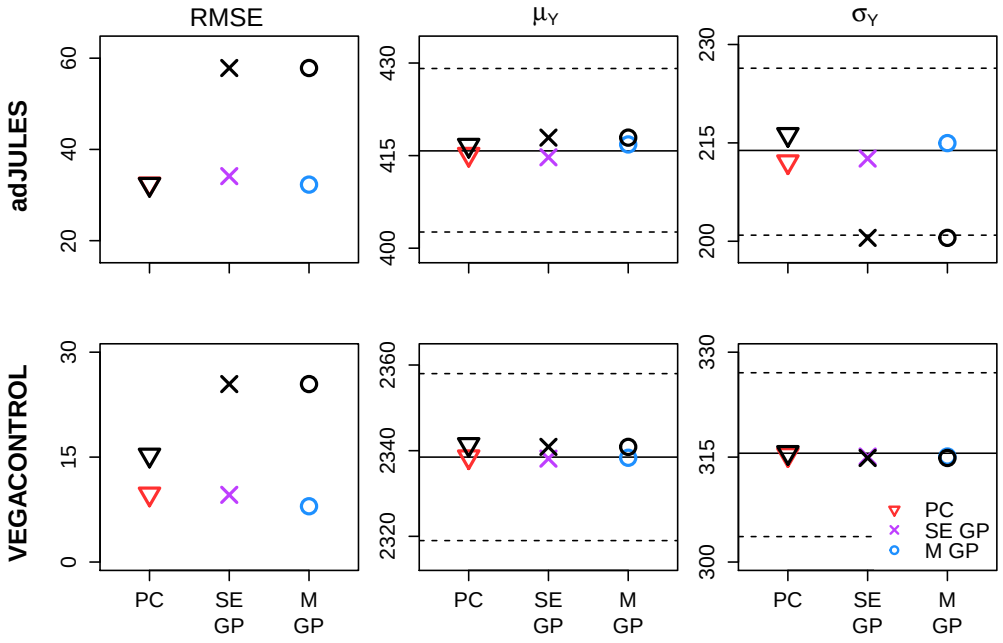


Figure 4.6.: Difference in root mean square error (RMSE), mean and standard deviation validation metrics when surrogates are built on large Sobol sequence designs instead of tensor grids. Sobol sequences of size 625 and 1024 were used in the adJULES (top panels) and VEGACONTROL (bottom panels) experiments respectively, to match the largest tensor grid designs. Validation metrics from the tensor grid designs are shown in black, whereas metrics from the Sobol sequence designs are shown in colour.



emulators built on the large Sobol sequence design become more accurate in terms of the RMSE metric and now have similar results to PC. While the estimates of the mean remain accurate for all surrogate methods with the change of design, for the standard deviation the GP emulator estimates are now within the confidence intervals. Validation metrics for PC do not worsen with the change in design type and remain highly accurate. It is expected these results would carry across to the exceedance probability and PDF validation metrics, but these experiments are not carried out here.

In summary, GP emulator methods (with either correlation function) are preferred for class 1 designs regardless of validation metric; RMSE results are similar for surrogates built on class 2 designs but PC is favoured for class 3 designs; all surrogate methods accurately estimate the simulator mean for class 2 and class 3 designs; PC is narrowly favoured for estimating the simulator standard deviation in all cases except for class 1 designs. For the exceedance probability metrics, neither method is accurate for class 1 designs but GP methods are preferred; PC is more accurate for class 2 designs; surrogate methods perform similarly for class 3 designs with a narrow preference for PC. Probability density function estimation results show that both methods give accurate results for large design sizes in classes 2 and 3; some preferences can be found for PC in the smaller designs in these classes. Finally, it

is observed that the performance gap between PC and GP methods built on large tensor grid designs can largely be eliminated with a switch to a Sobol sequence design of the same size.

4.4.2. VEGACONTROL experiment

Validation results from fitting PC, SE GP and M GP surrogates to the `VEGACONTROL` simulator are presented in Figures 4.6, 4.7, 4.8 and 4.9. Firstly, consider results for the RMSE, mean and standard deviation validation metrics (given by Equations (4.10)–(4.12)) shown in Figure 4.7. Once again for class 1 designs, both GP emulators outperform the PC surrogates regardless of the validation metric. In particular, the PC surrogate built on the second design of this class (size 21) has a large error for all validation metrics. For class 2 designs, all the surrogate approaches exhibit similar accuracy. This is especially the case when estimating the simulator mean and standard deviation, as all surrogate estimates are within the calculated confidence intervals. In this case, none of the surrogate methods would be preferred. However, with regards to the RMSE metric there is some evidence for favouring GP emulation over PC. While PC and SE GP surrogates have similar accuracy for all design sizes, the M GP surrogate consistently does better. This is one of the few cases in the experiments where the choice of correlation function for the GP emulator makes a substantial difference, and perhaps the Matérn correlation function is particularly suited to the output of the `VEGACONTROL` simulator. The converse is found for class 3 designs. While again all surrogate estimates of the simulator mean and standard deviation are of high quality (and almost identical in this case), there is some evidence to suggest that the PC surrogates perform better in terms of the RMSE metric. However, this can only be observed for the larger design in this class; for the two smaller designs all surrogates approaches have similar RMSE.

Secondly, consider the exceedance probability metrics (given by Equation (4.13)) presented in Figure 4.8. Straight away it can be seen that the surrogates perform better here than for the `adJULES` experiment, with many of the estimates falling within the calculated confidence intervals. This can be attributed to the difference in shape of the `adJULES` and `VEGACONTROL` simulator output PDFs. Comparing Figure 4.5 to Figure 4.9, it is evident that the `adJULES` simulator output has a much longer tail than the `VEGACONTROL` simulator output, which may be harder to estimate using a surrogate. This would directly affect the surrogates' ability to estimate the probability of exceeding two or three standard deviations above the mean. Returning to Figure 4.8, once again for class 1 designs the PC estimates are inaccurate and unstable to increases in design size, and GP methods are favoured. For class 2 and 3 designs, estimates from GP and PC surrogates are similar and of

Figure 4.7.: Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the VEGACONTROL simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines).

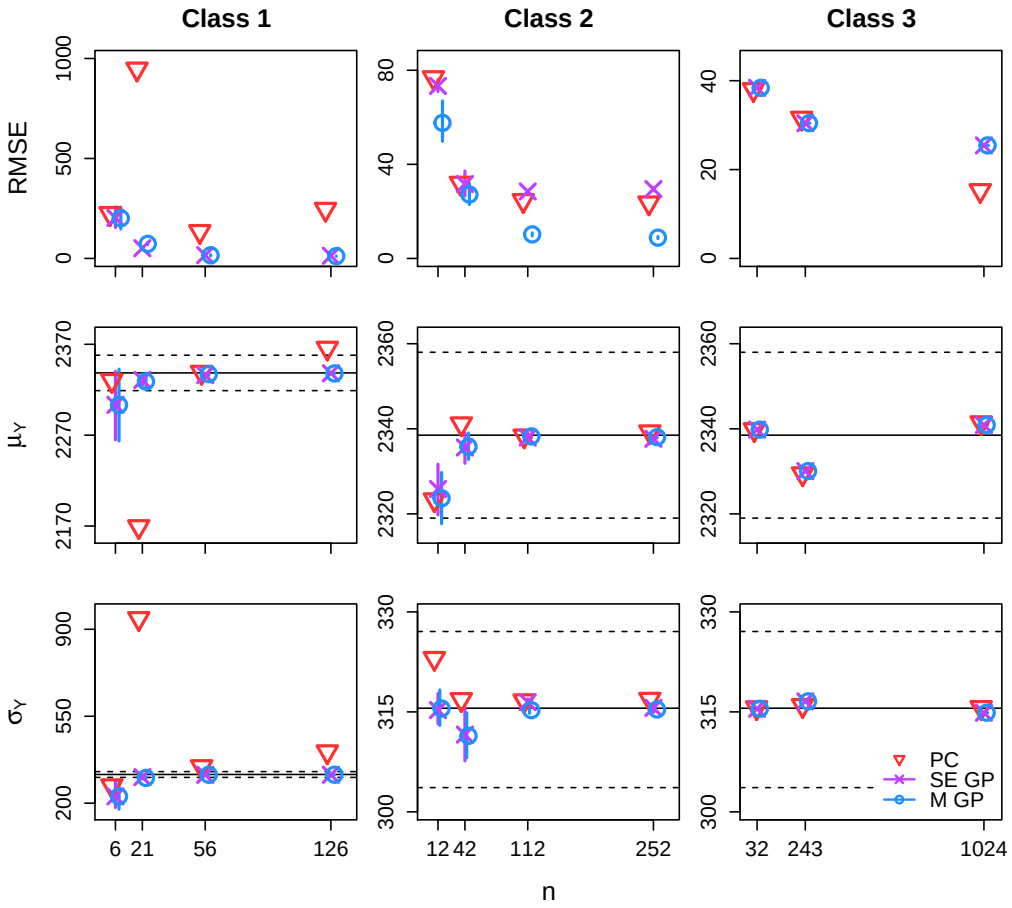
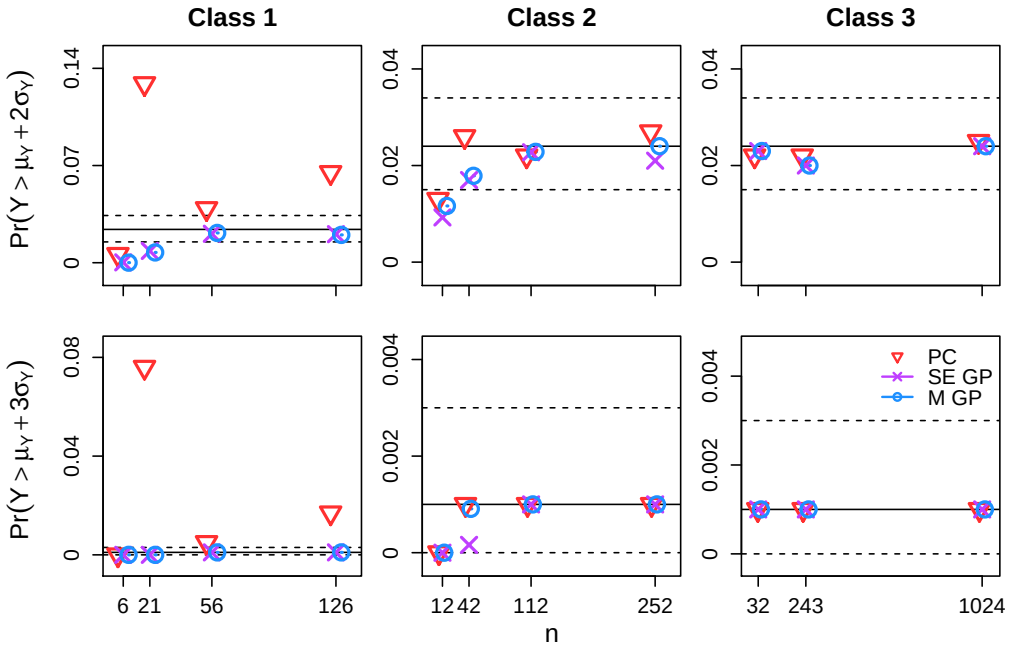


Figure 4.8.: Exceedance probabilities $Pr(Y > \mu_Y + 2\sigma_Y)$ and $Pr(Y > \mu_Y + 3\sigma_Y)$ as a function of design size n and design class for the VEGACONTROL simulator. The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines).



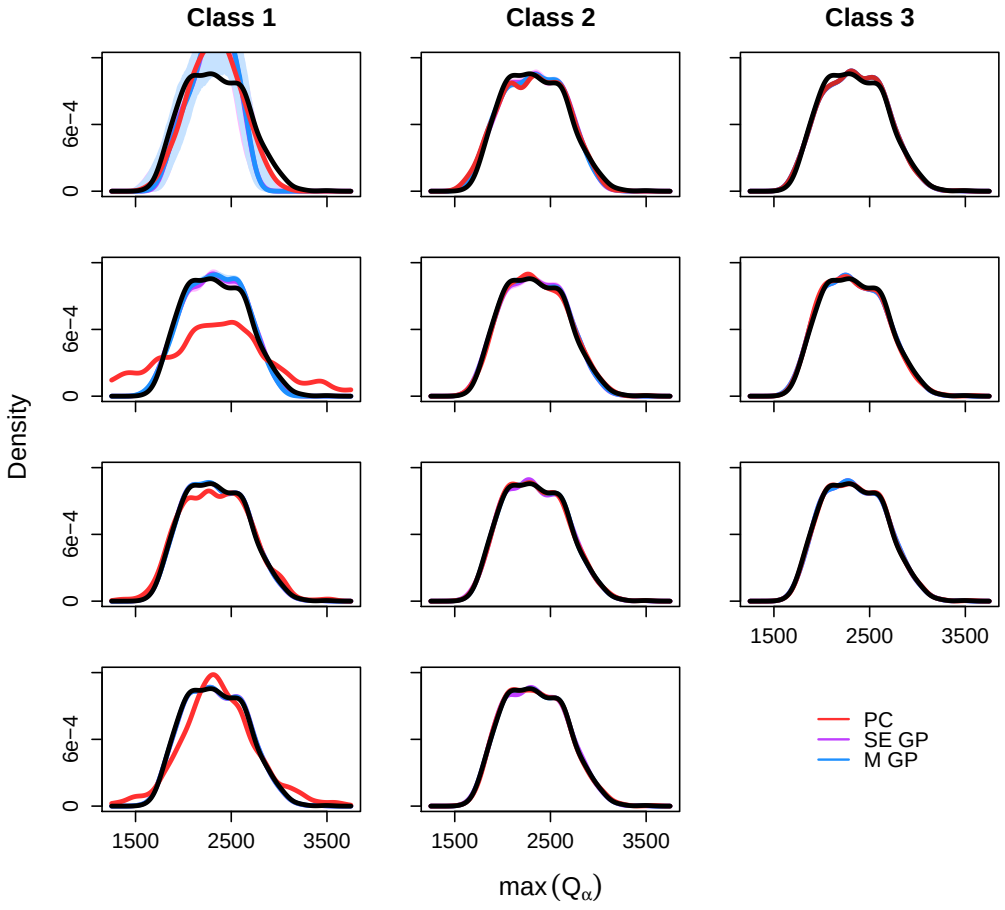
high quality for both exceedance probabilities and all design sizes considered. In this case, none of the surrogate methods would be preferred.

Now consider the PDF validation metrics shown in Figure 4.9. A clear preference for GP methods can be found for class 1 designs, where PC methods struggle to accurately estimate the PDF in the majority of cases. For class 2 and 3 designs, all surrogate estimates of the PDF are of high quality and there is little to choose between them.

Finally, PC, SE GP and M GP surrogates were also built on a Sobol sequence design of size 1024 for the `VEGACONTROL` simulator, and results for the RMSE, mean and standard deviation validation metrics are presented in Figure 4.6 (bottom panels). Similar to the conclusions drawn from the `adJULES` experiment, it is observed that GP emulators become more accurate in terms of the RMSE metric with the use of a large Sobol sequence design. However, in this case, the PC surrogate also benefits from a change in design type and has a lower RMSE, similar to that of the modified GP emulators. Estimates of the simulator mean and standard deviation remain accurate for all surrogates and are robust under the change in design type. Again, it is expected that these results would carry over to the exceedance probability and PDF validation metrics, but these experiments are not carried out here.

In summary, GP methods are preferred for class 1 designs regardless of the validation

Figure 4.9.: Probability density function (PDF) validation metrics for the VEGA-CONTROL simulator. Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates.



metric; mean, standard deviation, exceedance probability and PDF estimates are of a high quality and similar between surrogate approaches for class 2 and 3 designs; for a lower RMSE the M GP emulator is preferred for class 2 designs, whereas PC is favoured for class 3 designs. In general, the results from the `adJULES` experiments are replicated for the `VEGACONTROL` simulator, apart from the fact that all surrogates tended to calculate the validation metrics to a higher degree of accuracy for the latter case.

4.4.3. Discussion

Based on the validation results presented for the `adJULES` and `VEGACONTROL` simulators in the previous two sections, it is clear that one surrogate methodology does not unanimously outperform the other, but that the best method out of GP and PC depends on the modelling goals of the practitioner as well as the type and size of the experimental design. If the desire is for an accurate surrogate across the input space — which here is measured using the RMSE metric — broadly speaking GP emulators are preferred for Sobol sequence designs, whereas PC surrogates are favoured for tensor grid designs. There are two possible reasons for why this should be. Firstly, research has found (for example, Urban and Fricker (2010)) that GP emulators are more accurate when built on less structured designs than tensor grids. This is mainly due to the collapsing property of tensor grid designs, where design points are repeated if the design is projected onto a lower dimension. This does not happen for Sobol sequence or Latin Hypercube designs, and the extra data in each dimension tends to result in better estimation of the GP hyperparameters. Furthermore, tensor grids naturally have a smaller set of distances between points than less structured designs, and this is detrimental to the estimation of correlation length hyperparameters (recall that the correlation function only depends on the distance between two input settings). These effects were demonstrated in Section 3.3.3; for example, the 'banding' of the standard deviation in Figure 3.13. Secondly, the regression approach used for PC on Sobol sequence leads to surrogates which do not necessarily interpolate the simulator output, but the quadrature approach on tensor grids does. This effect may lead to a poorer performance in terms of RMSE. This has parallels with the use of the nugget parameter in GP emulation (Andrianakis and Challenor, 2012), where the interpolation property of the emulator is relaxed. Insight may also be gained by using the property given in Section 3.4 of Chapter 3: that regression PC is a special case of GP emulation when the mean function is a truncated PCE and the correlation length hyperparameters $\boldsymbol{\delta} \to \mathbf{0}$. The extra information provided by the estimation of the correlation length hyperparameters of the GP emulator may lead to better performance in terms of the RMSE metric.

If instead some statistical summaries of the simulator output are required, there is a slightly different focus in selecting the best surrogate approach. When estimating the simulator mean, results showed that both GP and PC surrogates gave very similar and highly accurate estimates for both design types, even for small design sizes. This was also the case for estimating the standard deviation for the `VEGACONTROL` simulator, but a narrow preference for PC could be found in the `adJULES` experiment. When estimating the probability of exceeding two or three standard deviations above the simulator mean, surrogate estimates were similar and highly accurate for the `VEGACONTROL` simulator, but small preferences for PC could be seen for the `adJULES` simulator (particularly for class 2 designs). Finally, when estimating the PDF of the simulator output, there was little difference between the surrogate approaches.

For all validation metrics, a clear preference for GP emulators was evident for class 1 designs. This suggests that GP emulators are more accurate for very small design sizes. The results also align with those given in Hosder et al. (2007), who suggested that the uniquely-determined regression technique for PC should be avoided, with at least twice over-determined regression giving more stable behaviour. However, in the case of small design sizes for PC, the requirement of at least twice over-determined regression means that the truncation order of the PC surrogate must be reduced to compensate. Alternatively, more advanced methods such as sparse polynomial chaos (Blatman and Sudret, 2011) could be implemented for improved stability, which use regularisation to reduce the number of terms in the expansion (similar to the basis function selection used for selecting the GP emulator mean functions).

Lastly, given the performance gap between PC and GP methods for the largest tensor grid designs, it was investigated how the validation metrics changed when the surrogates were built on a Sobol sequence design of the same size. In short, it was observed that the performance gap could be closed completely with this change of design. In particular, the RMSE for GP emulators greatly reduced to be consistent with polynomial chaos. This reiterates how important design choice is in building a surrogate. If a large design size is possible, results from these experiments suggest that the PC surrogates should be built on tensor grid designs, whereas GP emulators should be built on Sobol sequence designs (or an alternative such as a Latin Hypercube).

4.5. General advantages and disadvantages

The experiments outlined in Sections 4.3 and 4.4, as well as the toy examples given in Sections 3.2.3 and 3.3.3, have highlighted a number of points concerning the

computational cost, flexibility and practicality of the two surrogate methodologies. In this section, comparative advantages and disadvantages of PC and GP surrogate approaches for each of these points are given, in the effort to display the benefits and shortcomings of each method.

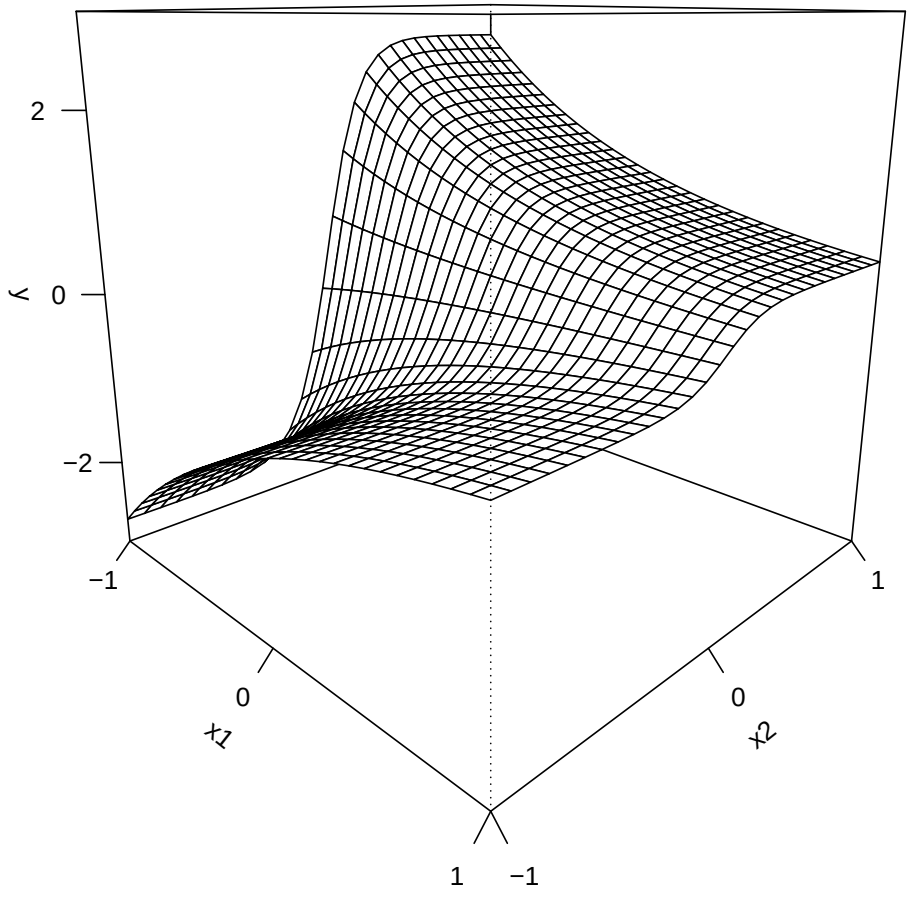
In terms of the computational cost of building the surrogate, for PC most of the work is done prior to the experiment in choosing the correct polynomial basis to be used in the expansion. Recall that this depends solely on the probability distributions of the simulator inputs. The subsequent non-intrusive estimation of the expansion coefficients is generally cheap, requiring the evaluation of plug-in formulas (NISP using quadrature) or simple matrix algebra (regression). Therefore, if a simple UQ analysis such as the estimation of the mean or standard deviation of the simulator output is all that is required, PC would be the faster approach. Conversely, most of the computational cost comes from fitting the GP emulator itself, that is, estimating the hyperparameters. This is an expensive operation which scales as $\mathcal{O}(n^3)$. Moreover, problems can arise when estimating the correlation lengths through maximisation of the likelihood, and the optimisation may have to be repeated several times to obtain the global maximum. However, these problems usually have a simple fix (for example, with the addition of a nugget parameter (Andrianakis and Challenor, 2012)).

Regarding the comparative flexibility of the surrogate approaches to adapt to different scenarios, it is clear that GP emulators have the edge. In the experiments outlined in Section 4.3, the sizes and types of the experimental design were restricted by PC; whether it be choosing Gaussian quadrature rules for the tensor grid designs, or restricting the size of design to ensure a specific truncation order for the PC surrogate. In this sense, a subtle advantage was given to the PC approach. In contrast, there are no real restrictions on the size and type of design to build a GP emulator. The fact that the GP emulators performed as well as PC surrogates based on the design choices is testament to the flexibility of the approach. It must also be remarked that although many simulators can be quickly and accurately approximated using a polynomial function, some are bound to exhibit more complicated behaviour. There is no doubt that GP emulators can deal with a wider range of simulator behaviours, and this can be done relatively easily with changes in the mean and covariance functions. To add more weight to this argument, recall the two-dimensional simulator given in Chapter 3:

$$y = \eta(x_1, x_2) = \exp(-x_1)\tanh(5x_2), \qquad X_1, X_2 \sim \text{Unif}[-1, 1]. \tag{4.14}$$

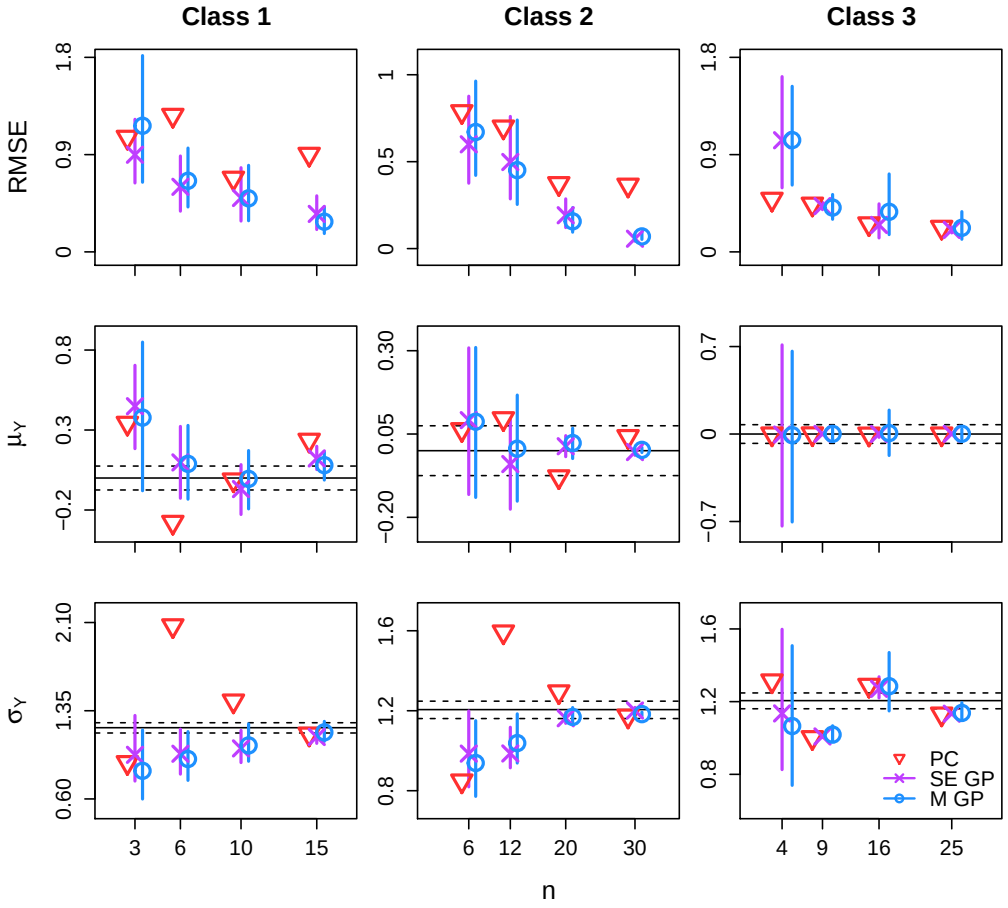
The simulator, plotted in Figure 4.10, is clearly a nonlinear function of the inputs. Therefore, it is expected that a low-order PC surrogate would perform poorly. The experiments outlined in Section 4.3 are repeated for this simulator, with the same

Figure 4.10.: Plot of the two-dimensional toy simulator in Equation (4.14) for $x_1, x_2 \in [-1, 1]$.



design rationale but for input dimension $d = 2$. Validation results from fitting PC, SE GP and M GP surrogates to the two-dimensional toy simulator are presented in Figures 4.11, 4.12 and 4.13. Interestingly, PC surrogates built on class 1 designs are not as poor as those for the `adJULES` and `VEGACONTROL` experiments. However, the two types of GP emulators still give more accurate and stable results, so they are preferred here. One of the most striking results is the difference between the two methods for class 2 designs. For all validation metrics and the majority of design sizes, GP emulators consistently perform better than PC surrogates. Note also that for class 3 designs, the performance gap between the two methods has now closed considerably. Finally, this nonlinear function has a bimodal PDF that all surrogate methods find difficult to estimate, especially for small design sizes. Nonetheless, a preference for GP emulators is still apparent. These results provide good evidence that GP emulators are more suitable for modelling nonlinear simulator behaviour. Nevertheless, PC surrogates still provide respectable accuracy for a fraction of the

Figure 4.11.: Root mean square error (RMSE), mean μ_Y and standard deviation σ_Y validation metrics as a function of design size n and design class for the two-dimensional toy simulator in Equation (4.14). The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator mean and standard deviation (solid black lines) are shown with 95% confidence intervals (dashed black lines).

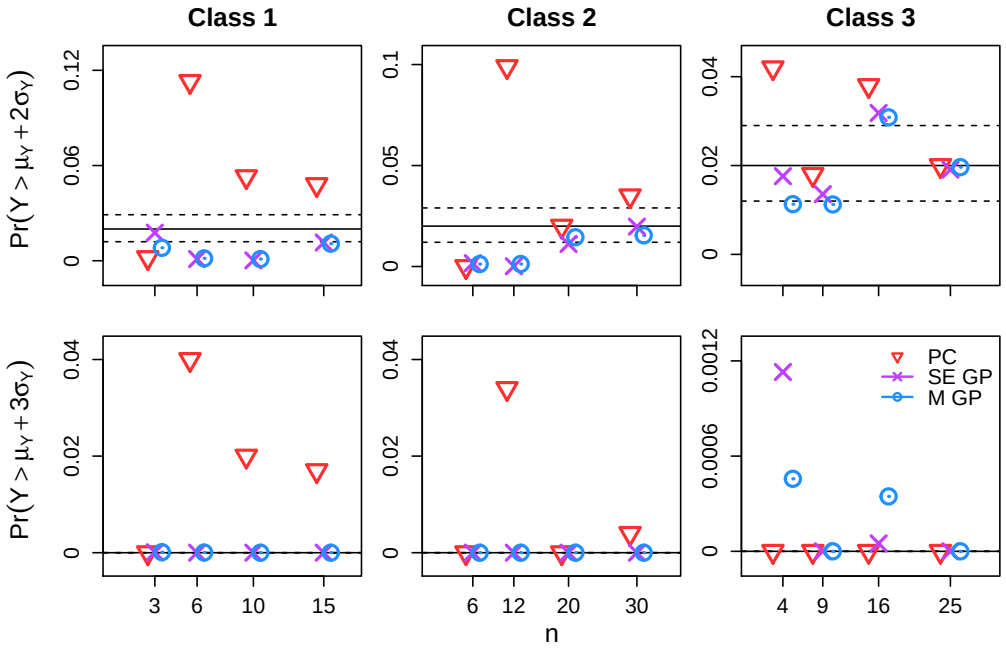


cost[3].

Finally, concerning the practicality of the two surrogate methodologies, it is clear that both approaches offer a fast approximation to the simulator at any untried input configuration. This is useful for subsequent Monte Carlo based UQ tasks such as sensitivity analysis or calibration of the simulator. Crucially however, GP emulators not only provide a single prediction but have the advantage of readily available uncertainty information due to the distributional assumptions. This not only allows the practitioner to evaluate where the emulator is most uncertain across the design space, but uncertainty can be fed through to various validation metrics (as shown in the above experiments). This is not the case for PC, and preference may be found for GP emulators if uncertainty information is required.

[3]The exact accuracy is undoubtedly related to how well a Taylor series would approximate the function, but this has not been considered in this work.

Figure 4.12.: Exceedance probabilities $Pr(Y > \mu_Y + 2\sigma_Y)$ and $Pr(Y > \mu_Y + 3\sigma_Y)$ as a function of design size n and design class for the two-dimensional toy simulator in Equation (4.14). The points have been jittered slightly for clarity. Gaussian process emulators have 95% confidence intervals about their estimates (solid lines). Simulator exceedance probabilities (solid black lines) are shown with 95% confidence intervals (dashed black lines).

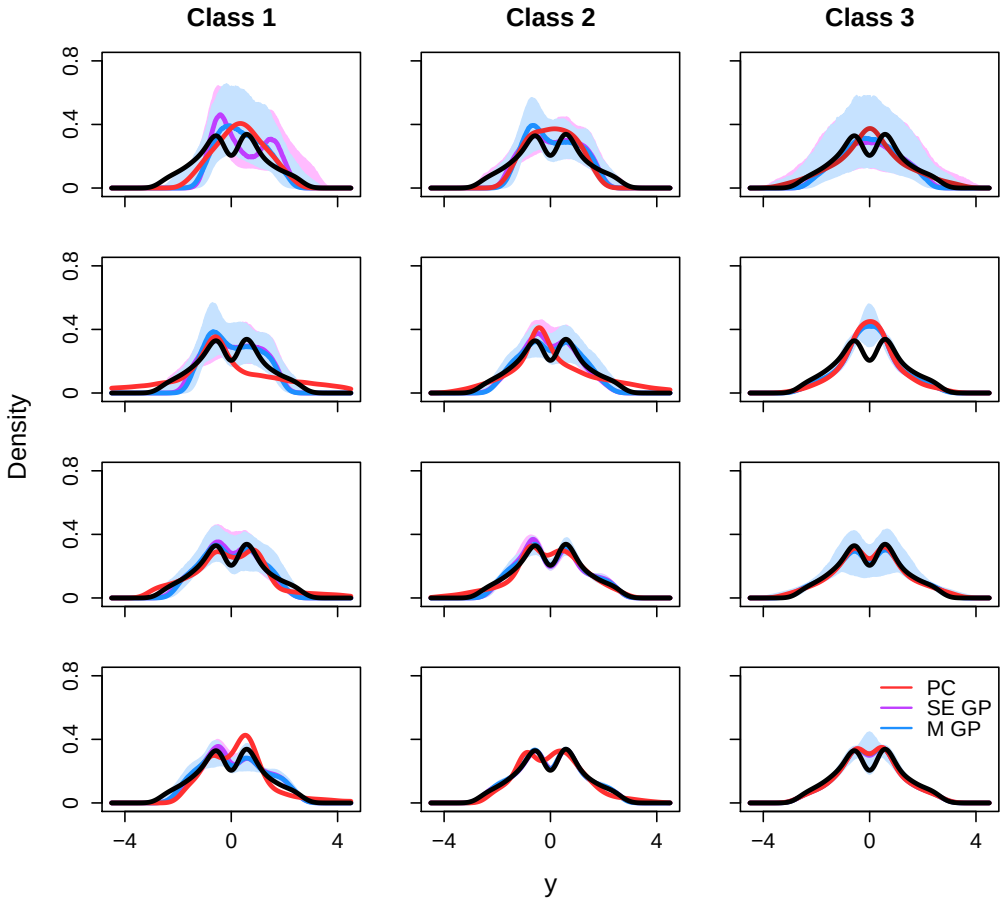


4.6. Conclusion

For surrogate-based approaches to uncertainty quantification for computer experiments, validation of the surrogate model is a vital stage in the analysis. If the validation is not carried out properly, any subsequent analysis performed using the surrogate towards an uncertainty quantification objective will not be valid. In Section 4.1 it was demonstrated that validation techniques for surrogate models generally fall into two categories: those based on additional validation runs of the simulator, and those which reuse the experimental design points in some way. Within these two categories however, the specific validation metric used to assess the quality of the surrogate or whether modelling assumptions have been satisfied are wide-ranging. With particular relevance to the two surrogate modelling approaches featured in this work, it was shown that validation methods for polynomial chaos and Gaussian process emulation are contrasting. This is perhaps one of the primary reasons why there has been a lack of research comparing the two approaches in different scenarios, in addition to the fact that the methods have been developed in their respective communities, with little communication between them.

To facilitate such a comparison, a set of simple validation metrics applicable to both PC and GP emulation were introduced in Section 4.2. The metrics were designed to assess the accuracy of the two surrogate methods by comparing their output to

Figure 4.13.: Probability density function (PDF) validation metrics for the two-dimensional toy simulator in Equation (4.14). Design classes are in columns and design size used to build surrogates increases further down the rows (see Table 4.1). The simulator output PDF is shown as a black line. Gaussian process emulators also have 95% confidence intervals about their estimates.



the corresponding simulator quantities evaluated at a validation design. Specifically, surrogates were used to estimate the mean, standard deviation, exceedance probabilities and the probability density function of the simulator output. The root mean square error also provided a single measure of surrogate fit. In Section 4.3, experiments were designed to examine how the comparative accuracy of the surrogates (measured using the validation metrics) changed with variations in the size and type of the experimental design used to build the surrogates.

The results presented in Section 4.4 for two simulators used in industry, named `adJULES` and `VEGACONTROL`, showed that one method did not unanimously outperform the other, but advantages can be gained in some cases, such that the preferred method depends on the modelling goals of the practitioner. Furthermore, the design type had considerable impact on which method was favoured. At the time of writing, the results from these experiments are the first example of a direct comparison of the performance of the two methods in the literature. The primary aim of the experiments was to provide useful advice to practitioners in UQ, namely which surrogate approach should be used in different modelling scenarios.

Finally, Section 4.5 outlined some general advantages and disadvantages of the two approaches which became apparent when carrying out the experiments. In particular, comments were made on the computational cost, flexibility and practicality of PC and GP emulation. A major disadvantage of PC is the fact that no uncertainty information is included, unlike GP emulators which are fully probabilistic surrogates. Despite providing efficient surrogate-based UQ, code uncertainty is not quantified in PC. This is addressed in Chapter 5, where a hybrid approach combining PC and GP emulation is presented.

5. Probabilistic numerics and polynomial chaos

In this chapter, a hybrid approach of Gaussian process emulation and polynomial chaos, named probabilistic polynomial chaos, is presented. The method draws upon an emerging scientific field known as probabilistic numerics, which treats classical numerical methods as statistical inference problems. The chapter is structured as follows. Section 5.1 concerns the field of probabilistic numerics. An introduction to probabilistic numerics and a short literature review of its recent developments are given in Section 5.1.1, with particular focus on probabilistic approaches to numerical integration. A relevant probabilistic integration method, called Bayesian quadrature, is outlined in Section 5.1.2, along with details on its implementation and some examples in Sections 5.1.3 and 5.1.4 respectively. The main focus of this chapter is the application of Bayesian quadrature to the non-intrusive spectral projection method for polynomial chaos, which was described and applied in Chapters 3 and 4 respectively. This novel combination of techniques gives rise to the probabilistic polynomial chaos methodology presented in Section 5.2. The mathematical details of probabilistic polynomial chaos are given in Section 5.2.1, followed by a technical discussion on its implementation in Section 5.2.2. The resulting probabilistic polynomial chaos surrogate is given in Section 5.2.3, and subsequently applied for a simple example in Section 5.2.4. The chapter is closed with a discussion and conclusion in Sections 5.3 and 5.4 respectively.

5.1. Probabilistic numerics

5.1.1. Introduction

Algorithms for numerical tasks such as linear algebra, integration, optimisation and solving ordinary or partial differential equations are the building blocks of modern scientific computation (Hennig et al., 2015). They are deterministic approaches which return point estimates for unknown quantities, based on a finite computational budget. For example, a quadrature rule estimates the value of an integral

by computing a weighted average of integrand evaluations at a set of points in the input domain. Clearly, an exact answer would require infinitely many evaluations. Since this is not possible due to time or computational restrictions, naturally all numerical methods introduce an error in their calculations. Many classical numerical methods come with a description of the error through theoretical convergence rates, but these are only satisfied under regularity conditions and are of little use in practice (Hennig et al., 2015). Even so, numerical algorithms often provide an estimate of the error at runtime which can be made as small as necessary with sufficient resources. A real issue arises when a chain of numerical methods are used in a computational pipeline, where numerical errors can propagate and accumulate with serious consequences (Cockayne et al., 2017).

Probabilistic numerics (Hennig et al., 2015) is an emerging research field in scientific computation, which argues that classical numerical methods should be treated as statistical inference problems. This is because numerical methods reason about latent quantities using data, with uncertainty arising from a lack of information in the solution of an intractable problem. Returning to the example of a quadrature rule, the value of the integral is a latent quantity, learned through observations of the integrand since the integral is not available in closed form. Numerical errors are often the only part of a statistical analysis for which uncertainty is not typically accounted for in a fully probabilistic way (Briol et al., 2015b). However, the premise of uncertainty in deterministic quantities is not new, and early discussions in the literature can be found in Poincaré (1912) and Erdös and Kac (1940). In particular, Hull and Swenson (1966) proposed simple statistical models for the propagation of rounding error for a class of differential equations. Foundations for a Bayesian approach to probabilistic numerics — now the standard approach — were laid by Diaconis (1988) and O'Hagan (1992).

Recently, probabilistic versions of several established numerical algorithms have been developed. These include numerical methods for the solution of ordinary differential equations (Skilling, 1992; Schober et al., 2014; Chkrebtii et al., 2016) and partial differential equations (Conrad et al., 2015; Owhadi, 2015), linear algebra (Hennig, 2015; Bartels et al., 2016) and optimisation (Hennig and Kiefel, 2013; Mahsereci and Hennig, 2015). However, the focus of this chapter is on probabilistic approaches for numerical integration, known as Bayesian quadrature (O'Hagan, 1991), Bayesian Monte Carlo (Rasmussen and Ghahramani, 2002) or probabilistic integration (Briol et al., 2015b). The methodology of Bayesian quadrature will be presented in Section 5.1.2, along with some issues for implementation and examples in Sections 5.1.3 and 5.1.4 respectively. In Section 5.2, it will be shown how Bayesian quadrature can be combined with the non-intrusive spectral projection method for polynomial chaos. This section is concluded with a short literature review of Bayesian quadrature

developments.

Consider the integral:

$$\mathcal{I} = \int_{\mathcal{X}} g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{5.1}$$

that is, the integral (expectation) of a function $g(\mathbf{x})$ with respect to a probability distribution $f_{\mathbf{X}}(\mathbf{x})$ on support $\mathcal{X}$. Similar to the discussions in the preceding chapters, it is assumed that the function $g(\mathbf{x})$ is expensive to evaluate, meaning that only a finite number of runs can be obtained. Bayesian quadrature proceeds by placing a prior on $g(\mathbf{x})$, and hence the value of the integral, $\mathcal{I}$. By evaluating the function at a set of input locations, prior information can be turned into a posterior using Bayes' rule. In particular, the posterior distribution for $\mathcal{I}$ is a quantification of the numerical error on the value of the integral, induced by finite evaluations of the integrand. The maximum a posteriori estimate can then be used as a point estimate for the value of the integral. O'Hagan (1991) considered the case where $f_{\mathbf{X}}(\mathbf{x})$ is a multivariate Gaussian distribution, and proposed a Gaussian process with a squared exponential correlation function as a prior for $g(\mathbf{x})$. Under these assumptions, derivation of the posteriors is analytical. The mathematical details of this method, which is known specifically as Bayes-Hermite quadrature, will be given in Section 5.1.2.

The original method proposed by O'Hagan (1991) has since been studied and extended in a number of ways. While frequentist approaches can be found in the literature — for example, Kong et al. (2003) — recent contributions have almost exclusively been Bayesian. Kennedy (1998) provided functionality for the probability distribution $f_{\mathbf{X}}(\mathbf{x})$ to be a mixture of Gaussian distributions or a family of skewed distributions. Several authors (Diaconis, 1988; Minka, 2000; Särkkä et al., 2016) have shown that classical quadrature rules (for example, the trapezium and Gaussian quadrature rules) can be recast in a probabilistic framework with certain choices for the Gaussian process correlation function and the distribution $f_{\mathbf{X}}(\mathbf{x})$. Correlation function and distribution pairs for which Bayesian quadrature is analytical were summarised by Briol et al. (2015b). These authors also studied the theoretical properties of Bayesian quadrature, deriving convergence rates that are faster than traditional methods under certain smoothness conditions, as well as proving that Bayesian quadrature posteriors contract on the true value of the integral in the asymptotic case. These features were also demonstrated empirically in earlier works, for example Rasmussen and Ghahramani (2002). Uncertainty in the Bayesian quadrature framework also allows for active learning of the integral — sequentially choosing the next integrand evaluation to minimise some criteria — and several algorithms have been proposed (Osborne et al., 2012a; Gunter et al., 2014; Briol et al., 2015a). Finally, Osborne et al. (2012b) adapted Bayesian quadrature for ratios of two integrals of the form in Equation (5.1) (typically found in Bayesian

computation), and Oates et al. (2016) extended Bayesian quadrature for the case when $f_{\mathbf{X}}(\mathbf{x})$ is not available in closed form.

5.1.2. Bayesian quadrature

In this section, a mathematical overview of the Bayesian quadrature (BQ) methodology presented by O'Hagan (1991) will be given. Integrals of the form in Equation (5.1) are considered here, but for reasons that will become clear in Section 5.2, assume that the value of the following more general integral is desired:

$$\mathcal{I} = \int_{\mathcal{X}} \mathbf{r}(\mathbf{x}) g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{5.2}$$

where $\mathbf{r}(\mathbf{x})$ is a vector of q_r known functions of $\mathbf{x}$. In this case, $\mathcal{I}$ is a vector with k-th element $\int_{\mathcal{X}} \mathbf{r}_k(\mathbf{x}) g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}$, $k = 1, \ldots, q_r$.

A classical quadrature rule estimates the value of the integral by computing:

$$\mathcal{I} \approx \sum_{i=1}^{n} w^{(i)} \mathbf{r}(\mathbf{x}^{(i)}) g(\mathbf{x}^{(i)}), \tag{5.3}$$

for weights $w^{(i)}$ and input settings $\mathbf{x}^{(i)}$, $i = 1, \ldots, n$, specified by the quadrature rule. The appropriate quadrature rule for a particular integral depends on the probability distribution $f_{\mathbf{X}}(\mathbf{x})$ and its support $\mathcal{X}$. As discussed in Section 5.1.1, numerical quadrature gives a deterministic estimate of $\mathcal{I}$ without accounting for the uncertainty arising from a finite number of evaluations of the integrand.

Instead, BQ casts numerical integration in a probabilistic framework by placing a prior on the class of functions for $g(\mathbf{x})$. In particular, a Gaussian process (GP) prior is used. Recall from Chapter 3 this has the form:

$$g(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(M(\mathbf{x}; \boldsymbol{\beta}), \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\right), \tag{5.4}$$

where the mean function is a regression model $M(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta}$ of q known basis functions, and the correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$ is stationary and separable. For the remainder of this chapter, the correlation function will be assumed to be of the separable squared exponential form:

$$C(\mathbf{x}, \mathbf{x}', \boldsymbol{\delta}) = \prod_{j=1}^{d} C(x_j, x_j'; \delta_j) = \prod_{j=1}^{d} \exp\left[-\frac{1}{2}\left(\frac{x_j - x_j'}{\delta_j}\right)^2\right]. \tag{5.5}$$

Note this is equivalent to the version given in Equation (3.35) in Chapter 3. The squared exponential correlation function is used in the original work of O'Hagan

(1991), and allows for the analytical derivation of the posteriors (but is not necessary).

Given this prior and a set of n function evaluations at an experimental design $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$, here denoted $\mathbf{g} = \left(g(\mathbf{x}^{(1)}), \ldots, g(\mathbf{x}^{(n)})\right)^\top$, recall that a posterior for $g(\mathbf{x}) \,|\, \mathbf{g}$ can be derived as:

$$g(\mathbf{x}) \,|\, \mathbf{g} \sim t_{n-q}\left(M^{**}(\mathbf{x}), \hat{\sigma}^2 C^{**}(\mathbf{x}, \mathbf{x}')\right), \tag{5.6}$$

where expressions for the posterior mean and correlation functions, $M^{**}(\mathbf{x})$ and $C^{**}(\mathbf{x}, \mathbf{x}')$ respectively, as well as estimates for the hyperparameters $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$, can be found in Chapter 3.

The main task of BQ is to determine the posterior distribution for the value of the integral given a finite number of evaluations of the integrand, that is, $\mathcal{I} \,|\, \mathbf{g}$. Given the GP prior and the fact that the integral in Equation (5.2) is a linear functional of $g(\mathbf{x})$, derivation of this posterior distribution is analytical. It can be shown (O'Hagan, 1991) that $\mathcal{I} \,|\, \mathbf{g}$ is a location-scale shifted, multivariate Student's-t distribution with $n - q$ degrees of freedom. Its posterior mean vector and covariance matrix are:

$$\mathbb{E}\left[\mathcal{I} \,|\, \mathbf{g}\right] = \mathbf{R}\hat{\boldsymbol{\beta}} + \mathbf{T}\mathbf{A}^{-1}(\mathbf{g} - \mathbf{H}\hat{\boldsymbol{\beta}}), \qquad n - q > 1, \tag{5.7}$$

$$\operatorname{var}\left[\mathcal{I} \,|\, \mathbf{g}\right] = \hat{\sigma}^2 \mathbf{V}, \qquad n - q > 2, \tag{5.8}$$

where $\mathbf{H}$ is defined as

$$\mathbf{H} = \left(\mathbf{h}(\mathbf{x}^{(1)})^\top, \ldots, \mathbf{h}(\mathbf{x}^{(n)})^\top\right)^\top,$$

and $\mathbf{A}$ has elements

$$\mathbf{A}_{i,j} = C(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}; \boldsymbol{\delta}), \quad i, j = 1, \ldots, n.$$

These were originally defined in Equations (3.39) and (3.40) respectively in Chapter 3. Furthermore, $\mathbf{V}$ is a correlation matrix defined as:

$$\mathbf{V} = \mathbf{U} - \mathbf{T}\mathbf{A}^{-1}\mathbf{T}^\top + (\mathbf{R} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^\top\mathbf{A}^{-1}\mathbf{H})^{-1}(\mathbf{R} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H})^\top. \tag{5.9}$$

In words, $\mathbf{V}$ is a matrix containing the correlations between the q_r elements of $\mathcal{I}$ given the function evaluations $\mathbf{g}$.

The BQ mean in Equation (5.7) can be used as a point estimate for $\mathcal{I}$, whereas the BQ variance in Equation (5.8) can be used to quantify the uncertainty due to only evaluating the integrand n times. Note the important restriction on the degrees of freedom for the Student's-t distribution for its mean and variance to be defined. To obtain uncertainty information, given by the variance in Equation (5.8), the degrees

of freedom must satisfy $n - q > 2$.

Both Equation (5.7) and (5.8) rely on the matrices $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$, which themselves are integrals, defined as:

$$\begin{aligned}
\mathbf{R} &= \mathbb{E}\left[\mathbf{r}(\mathbf{x})\mathbf{h}(\mathbf{x})^\top\right] \\
&= \int_{\mathcal{X}} \mathbf{r}(\mathbf{x})\mathbf{h}(\mathbf{x})^\top f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x}, && (5.10)\\
\mathbf{T} &= \mathbb{E}\left[\mathbf{r}(\mathbf{x})\mathbf{t}(\mathbf{x})^\top\right] \\
&= \int_{\mathcal{X}} \mathbf{r}(\mathbf{x})\mathbf{t}(\mathbf{x})^\top f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x}, && (5.11)\\
\mathbf{U} &= \mathbb{E}\left[\mathbb{E}\left[C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})\mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x}')^\top\right]\right] \\
&= \int_{\mathcal{X}}\int_{\mathcal{X}} C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})\mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x}')^\top f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}'. && (5.12)
\end{aligned}$$

The $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ matrices can be described as follows. The matrix $\mathbf{R}$ is the expectation of the vector $\mathbf{r}(\mathbf{x})$ and the basis functions $\mathbf{h}(\mathbf{x})$ of the GP mean function. That is, it is a $q_r \times q$ matrix containing the integrals of the q_r functions of $\mathbf{r}(\mathbf{x})$ multiplied by the q basis functions of $\mathbf{h}(\mathbf{x})$. The matrix $\mathbf{T}$ is the expectation of the vector $\mathbf{r}(\mathbf{x})$ and the vector of correlations between $\mathbf{x}$ and the n experimental design points in $\mathcal{D}$, defined as (see Equation (3.44) in Chapter 3):

$$\mathbf{t}(\mathbf{x})^\top = \left(C(\mathbf{x},\mathbf{x}^{(1)};\boldsymbol{\delta}),\ldots,C(\mathbf{x},\mathbf{x}^{(n)};\boldsymbol{\delta})\right).$$

Alternatively, $\mathbf{T}$ is a $q_r \times n$ matrix containing the integrals of the q_r functions of $\mathbf{r}(\mathbf{x})$ multiplied by the n correlations in $\mathbf{t}(\mathbf{x})^\top$. The matrix $\mathbf{U}$ is the double expectation of the vector $\mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x}')^\top$ and the GP correlation function $C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})$, over all combinations of $\mathbf{x}$ and $\mathbf{x}'$ in the support $\mathcal{X}$. In other words, $\mathbf{U}$ is a $q_r \times q_r$ matrix containing the double integrals of the q_r functions of $\mathbf{r}(\mathbf{x})$ multiplied by the correlation function $C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})$. All expectations (integrals) are taken with respect to the probability distribution $f_{\mathbf{X}}(\mathbf{x})$.

It is important for the integrals in Equations (5.10)–(5.12) to have analytical solutions, otherwise the probabilistic integration of Equation (5.2) using BQ becomes an infinite nested chain of numerical integration problems. Several authors have shown that analytical solutions are possible with certain choices for the basis functions $\mathbf{h}(\mathbf{x})$, the correlation function $C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})$, and the probability distribution $f_{\mathbf{X}}(\mathbf{x})$ (for example, Briol et al. (2015b)). As already stated, the squared exponential correlation function in Equation (5.5) is used here. Furthermore, the d-dimensional probability distribution $f_{\mathbf{X}}(\mathbf{x})$ will be assumed to be the product of d independent

and identically distributed standard Gaussian distributions, that is:

$$\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x}) = \prod_{j=1}^{d} \frac{e^{-x_j^2/2}}{\sqrt{2\pi}}, \tag{5.13}$$

on a support $\mathcal{X} = \prod_{j=1}^{d} \mathcal{X}_j = (-\infty, \infty)^d$. This assumption is also made by O'Hagan (1991), and will be satisfactory for the remainder of this chapter. The possibility of extending the methodology for different probability distributions is discussed in Section 5.3.

As a final note, placing a GP prior on the function $g(\mathbf{x})$ induces the following multivariate Gaussian prior distribution for $\mathcal{I}$:

$$\mathcal{I} \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim MVN(\mathbf{R}\boldsymbol{\beta}, \sigma^2\mathbf{U}). \tag{5.14}$$

5.1.3. Implementation

In this section, two issues affecting the implementation of the BQ methodology presented in Section 5.1.2 are discussed: the analytical solutions of the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals and experimental design.

Analytical solutions of R, T and U

Consider the solutions of the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals in Equations (5.10)–(5.12) respectively. These integrals have analytical solutions for certain choices of the correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$, probability distribution $f_{\mathbf{X}}(\mathbf{x})$, basis functions $\mathbf{h}(\mathbf{x})$ and vector $\mathbf{r}(\mathbf{x})$. As already discussed, here it is assumed that the GP correlation function is of the separable squared exponential form in Equation (5.5), and the probability distribution is the product of d standard Gaussian distributions as given by Equation (5.13). Alternative choices for the correlation function and distribution for which the solution of $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ remains tractable are given by Briol et al. (2015b). Moreover, it is assumed that elements of the $\mathbf{r}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ vectors are separable, that is, their k-th elements can be written as:

$$\mathbf{r}_k(\mathbf{x}) = \prod_{j=1}^{d} r_k(x_j), \tag{5.15}$$

$$\mathbf{h}_k(\mathbf{x})^\top = \prod_{j=1}^{d} h_k(x_j). \tag{5.16}$$

The separability of the correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$, probability distribution $f_{\mathbf{X}}(\mathbf{x})$, as well as the vectors $\mathbf{r}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$, means that the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals can be

written as the product of d univariate integrals. Specifically, the k,l-th elements of $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ can be expressed as follows:

$$\begin{aligned}\mathbf{R}_{k,l} &= \int_{\mathcal{X}} \mathbf{r}_k(\mathbf{x})\mathbf{h}_l(\mathbf{x})^\top f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \prod_{j=1}^{d}\left\{\int_{\mathcal{X}_j} r_k(x_j)h_l(x_j)f_{X_j}(x_j)\,\mathrm{d}x_j\right\} \\ &= \prod_{j=1}^{d}\left\{\int_{-\infty}^{\infty} r_k(x_j)h_l(x_j)\frac{e^{-x_j^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x_j\right\} \\ &= \prod_{j=1}^{d} R_{k,l}^{(j)}, \qquad k=1,\ldots,q_r,\, l=1,\ldots,q, \end{aligned} \tag{5.17}$$

$$\begin{aligned}\mathbf{T}_{k,l} &= \int_{\mathcal{X}} \mathbf{r}_k(\mathbf{x})C(\mathbf{x},\mathbf{x}^{(l)};\boldsymbol{\delta})f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \prod_{j=1}^{d}\left\{\int_{\mathcal{X}_j} r_k(x_j)C(x_j,x_j^{(l)};\delta_j)f_{X_j}(x_j)\,\mathrm{d}x_j\right\} \\ &= \prod_{j=1}^{d}\left\{\int_{-\infty}^{\infty} r_k(x_j)\exp\left[-\frac{1}{2}\left(\frac{x_j-x_j^{(l)}}{\delta_j}\right)^2\right]\frac{e^{-x_j^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x_j\right\} \\ &= \prod_{j=1}^{d} T_{k,l}^{(j)}, \qquad k=1,\ldots,q_r,\, l=1,\ldots,n, \end{aligned} \tag{5.18}$$

$$\begin{aligned}\mathbf{U}_{k,l} &= \int_{\mathcal{X}}\int_{\mathcal{X}} C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})\mathbf{r}_k(\mathbf{x})\mathbf{r}_l(\mathbf{x}')f_{\mathbf{X}}(\mathbf{x})f_{\mathbf{X}}(\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \\ &= \prod_{j=1}^{d}\left\{\int_{\mathcal{X}_j}\int_{\mathcal{X}_j} C(x_j,x_j';\delta_j)r_k(x_j)r_l(x_j')f_{X_j}(x_j)f_{X_j}(x_j')\,\mathrm{d}x_j\,\mathrm{d}x_j'\right\} \\ &= \prod_{j=1}^{d}\left\{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp\left[-\frac{1}{2}\left(\frac{x_j-x_j'}{\delta_j}\right)^2\right] r_k(x_j)r_l(x_j')\frac{e^{-x_j^2/2}}{\sqrt{2\pi}}\frac{e^{-x_j'^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x_j\,\mathrm{d}x_j'\right\} \\ &= \prod_{j=1}^{d} U_{k,l}^{(j)}, \qquad k=1,\ldots,q_r,\, l=1,\ldots,q_r. \end{aligned} \tag{5.19}$$

Analytical solutions to $R_{k,l}^{(j)}$, $T_{k,l}^{(j)}$ and $U_{k,l}^{(j)}$, defined by Equations (5.17)–(5.19), are now required for $j=1,\ldots,d$, and specific choices for $r_k(x_j)$ and $h_l(x_j)$. In principle, the $\mathbf{r}(\mathbf{x})$ vector can include any known functions of $\mathbf{x}$, provided that tractable solutions of $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ are still possible, but here it is assumed that $\mathbf{r}(\mathbf{x})=(1)$, so that $q_r=1$, without loss of generality (more complicated cases for $\mathbf{r}(\mathbf{x})$ are considered in Section 5.2.2). As a consequence, $\mathbf{R}$ and $\mathbf{T}$ are $1\times q$ and $1\times n$ vectors respectively, and $\mathbf{U}$ is a scalar. The analytical solutions of $U^{(j)}$, as well as the l-th elements $R_l^{(j)}$

and $T_l^{(j)}$ are now required, where:

$$R_l^{(j)} = \int_{-\infty}^{\infty} h_l(x_j) \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j, \qquad l = 1, \ldots, q, \tag{5.20}$$

$$T_l^{(j)} = \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2} \left(\frac{x_j - x_j^{(l)}}{\delta_j} \right)^2 \right] \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j, \qquad l = 1, \ldots, n, \tag{5.21}$$

$$U^{(j)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2} \left(\frac{x_j - x_j'}{\delta_j} \right)^2 \right] \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \frac{e^{-x_j'^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j \, \mathrm{d}x_j'. \tag{5.22}$$

Firstly, the solution of $R_l^{(j)}$ in Equation (5.20) will be discussed. Note that $R_l^{(j)} = \mathbb{E}\left[h_l(X_j)\right]$, that is, the expectation of the function $h_l(X_j)$ taken with respect to the standard Gaussian distribution. Recall that the vector $\mathbf{h}(\mathbf{x})$ contains basis functions for the mean function of the GP. Here, it is assumed that the individual basis functions in Equation (5.16) are of the form $h_l(X_j) = X_j^b$, $b \in \mathbb{N}_0$, as is often the case in the GP literature (O'Hagan, 2006). In this case, $R_l^{(j)}$ are the moments of the standard Gaussian distribution. It is well documented that (for example, Patel and Read (1996)), for the standard Gaussian distribution:

$$\mathbb{E}\left[X^b\right] = \begin{cases} 0 & b = 2n - 1 \text{ (odd)}, \\ DF(n) & b = 2n \text{ (even)}, \end{cases} \tag{5.23}$$

where $DF(n) = (2n-1)!! \equiv (2n-1) \times (2n-3) \times \ldots \times (3) \times (1)$, $n \in \mathbb{N}$, is a sequence defining the double factorials of odd numbers: $DF(n) = 1, 3, 5, 15, \ldots$. For convenience, $DF(0) = 1$ when $n = 0$. In this way, polynomial functions in $\mathbf{h}(\mathbf{x})$ will lead to the solution of $\mathbf{R}$ being product of double factorial numbers. For example, if $d = 3$ and $\mathbf{h}(\mathbf{x})^\top = (1 \quad x_1 x_2^3 x_3^5 \quad x_1^2 x_2^4 x_3^2)$, $\mathbf{R} = (1 \quad 0 \times 0 \times 0 \quad 1 \times 3 \times 1)$.

Secondly, the solution of $T_l^{(j)}$ in Equation (5.21) can be shown to be:

$$T_l^{(j)} = \frac{1}{\sqrt{1 + \frac{1}{\delta_j^2}}} \exp\left[-\frac{x_j^{(l)2}}{2(\delta_j^2 + 1)} \right], \qquad j = 1, \ldots, d, \; l = 1, \ldots, n. \tag{5.24}$$

Finally, the solution of $U^{(j)}$ in Equation (5.22) can be shown to be:

$$U^{(j)} = \frac{1}{\sqrt{1 + \frac{2}{\delta_j^2}}}, \qquad j = 1, \ldots, d. \tag{5.25}$$

More details on the derivation of $T_l^{(j)}$ and $U^{(j)}$ can be found in O'Hagan (1991) and Rasmussen and Ghahramani (2002).

Experimental design

Traditional quadrature rules restrict the experimental design points $\mathbf{x}^{(i)}$, $i = 1, \ldots, n$, to be at pre-specified locations. Similarly, the corresponding weights $w^{(i)}$, $i = 1, \ldots, n$, are specified by the quadrature rule. For example, a 3-point Gauss-Hermite quadrature rule in one-dimension uses the design $\mathcal{D} = (x^{(1)} = -1.732, x^{(2)} = 0, x^{(3)} = 1.732)$ and weights $\mathbf{w} = \left(w^{(1)} = 0.418, w^{(2)} = 1.671, w^{(3)} = 0.418\right)$ (to three decimal places). An n-point Gaussian quadrature rule is constructed to yield exact results when the integrand is a polynomial of degree $2n - 1$ or less.

In contrast, for BQ the restrictions on the experimental design are relaxed, provided that the design size n is large enough. Importantly, the design size $n > q + 2$ for the BQ posterior mean and variance to be defined — see Equations (5.7) and (5.8) for restrictions on the degrees of freedom for the Student's-t distribution. However, actually the design size must satisfy $n \geq q + d + 1$, since this is the number of hyperparameters to estimate in the GP emulator. As long as this is the case, the design points can be located anywhere in the input domain $\mathcal{X}$. This includes using traditional quadrature rule design points. Popular techniques such as Monte Carlo (MC) integration, which sample independently $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \sim f_{\mathbf{x}}(\mathbf{x})$, can now be recast in a Bayesian probabilistic framework by applying the BQ framework with the same sampling strategy. This technique has become known as Bayesian Monte Carlo (BMC) (Rasmussen and Ghahramani, 2002). Similarly, more advanced MC integration techniques such as quasi-Monte Carlo (QMC) or Markov Chain Monte Carlo (MCMC) integration can receive the same treatment, leading to Bayesian quasi-Monte Carlo (BQMC) and Bayesian Markov Chain Monte Carlo (BMCMC) approaches respectively (Briol et al., 2015b). In this way, BQ yields novel quadrature rules (different weights than traditional rules) by virtue of using different design strategies. Considering a quadrature rule as a weighted sum of function evaluations, $\mathcal{I} \approx \mathbf{w} \cdot \mathbf{g}$, for BQ the vector of weights is given as (O'Hagan, 1991):

$$\mathbf{w} = \mathbf{T}\mathbf{A}^{-1} + (\mathbf{R} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^{\top}\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\top}\mathbf{A}^{-1}. \tag{5.26}$$

Active or sequential sampling is also possible in the BQ framework. Given an initial experimental design $\mathcal{D}$, several algorithms have been proposed to select the next design point, $\mathbf{x}^{(n+1)}$, at which to evaluate the integrand, in order to minimise some criteria (Osborne et al., 2012a; Gunter et al., 2014; Briol et al., 2015a). The majority of algorithms choose $\mathbf{x}^{(n+1)}$ to minimise the posterior variance in Equation (5.8). For simplicity, active sampling will not be considered in this work.

5.1.4. Examples

In this section, a demonstrative example of BQ is given for a simple one-dimensional function. Consider solving the following integral:

$$\mathcal{I} = \int_{-\infty}^{\infty} g(x) \frac{e^{-x^2/2}}{\sqrt{2\pi}} \mathrm{d}x, \tag{5.27}$$

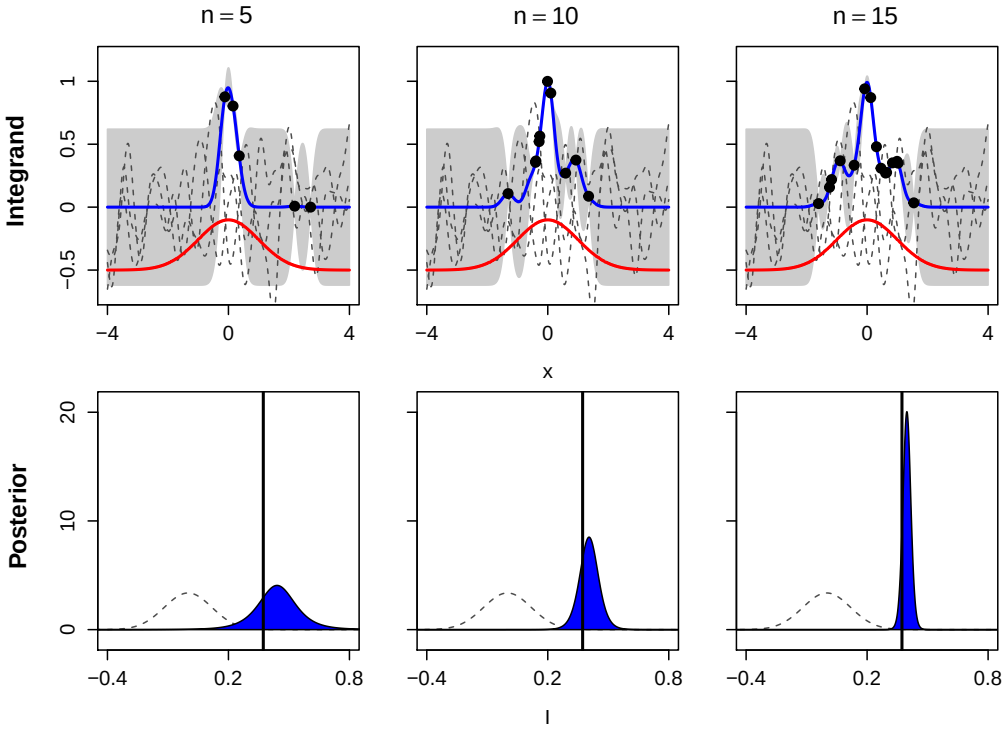
that is, the integral (expectation) of a function $g(x)$ with respect to the standard Gaussian distribution. Let $g(x) = \exp(-\sin^2(3x) - x^2)$[1]. A GP prior is placed on $g(x)$ of the form in Equation (5.4), with a constant mean and squared exponential correlation function. Instead of estimating the hyperparameters β, σ^2 and δ from data using the procedure outlined in Section 3.3.2, for simplicity these are fixed at the values $\beta = 0$, $\sigma^2 = 0.1$ and $\delta = 0.2$ here. A small nugget term of size 1×10^{-7} is also added to the diagonal elements of the correlation matrix to aid numerical conditioning when deriving the posterior. The focus of this example is comparing the performance of BQ under variations in the sampling strategy and design size. Specifically, MC and QMC samples are used with design sizes $n = 5, 10, 15$. QMC samples are generated using a Sobol sequence (Niederreiter, 1988).

Figures 5.1 and 5.2 show the performance of BQ with MC and QMC samples respectively. The top row of each figure concerns the integrand space, and presents GP prior draws and posterior fits for the function $g(x)$ given the samples. The bottom row of each figure demonstrates how this information is projected to the space of $\mathcal{I}$, the value of the integral, and prior and posterior distributions from BQ are shown. The true value of the integral, obtained from a large MC sample, is also plotted.

With the above values for the hyperparameters, the draws from the prior GP are highly variable (but still infinitely differentiable) functions centred around zero. The prior for $\mathcal{I}$ in Equation (5.14) can be calculated to be $N(0, 0.014)$, since $\mathbf{R} = (1)$ and $\mathbf{U} = (0.14)$, and is some way away from the true value of the integral. When introducing evaluations of the integrand, the posterior GP interpolates them (subject to the small nugget term), and provides a measure of uncertainty where the function has not yet been evaluated. Using the methodology of BQ presented in Section 5.1.2, the posterior for the integrand can be turned into a posterior for the value of the integral. For both MC and QMC samples, the mean of the posterior becomes closer to the true value of the integral as more design points are added, and the posterior variance reduces. The fact that the posterior contains the true value even for small sample sizes demonstrates the data-driven nature of the approach, where prior specification has little impact. As is the case for traditional numerical integration, the convergence rate of BQ for QMC samples is faster than that of the

[1] This function was also featured in Hennig et al. (2015).

Figure 5.1.: Bayesian quadrature for the one-dimensional example in Equation (5.27) using Monte Carlo samples of size $n = 5, 10, 15$ (columns). Top row: three draws from the GP prior for $g(x)$ (black dashed lines), posterior GP mean (blue line) with 95% credible intervals (grey shading) for $g(x)$ given the samples (black dots), standard Gaussian distribution density for reference (red line, not to scale). Bottom row: prior (black dashed line) and posterior (blue shading) distributions for $\mathcal{I}$ with true value (black vertical line).



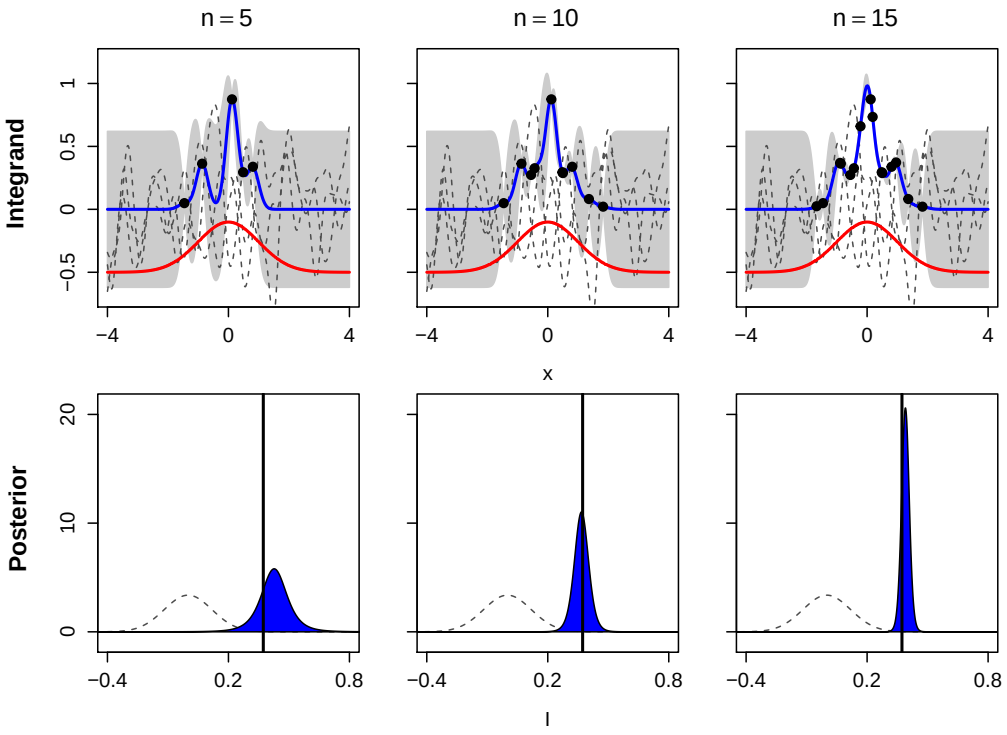
MC samples. This is due to more efficient spacing of the design points in the input domain.

5.2. Probabilistic polynomial chaos

In this section, a novel methodology called probabilistic polynomial chaos (PPC) will be described. The approach applies the BQ methodology outlined in Sections 5.1.2 and 5.1.3 to the non-intrusive spectral projection (NISP) method for polynomial chaos (PC). A recap of PC and the NISP method will now be given; a more in depth treatment can be found in Sections 3.2.1 and 3.2.2.

Recall an expensive, deterministic, black-box simulator, $y = \eta(\mathbf{x})$, with inputs $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and output $y \in \mathbb{R}$. It is assumed that the inputs can be modelled as a random vector $\mathbf{X}$ with known probability distribution $f_{\mathbf{X}}(\mathbf{x})$. Furthermore, it is assumed that the inputs are independent of one another, such that $f_{\mathbf{X}}(\mathbf{x}) = \prod_{j=1}^{d} f_{X_j}(x_j)$ on support $\mathcal{X} = \prod_{j=1}^{d} \mathcal{X}_j$. In PC, the induced uncertainty on the

Figure 5.2.: Bayesian quadrature for the one-dimensional example in Equation (5.27) using Quasi Monte Carlo samples of size $n = 5, 10, 15$ (columns). Top row: three draws from the GP prior for $g(x)$ (black dashed lines), posterior GP mean (blue line) with 95% credible intervals (grey shading) for $g(x)$ given the samples (black dots), standard Gaussian distribution density for reference (red line, not to scale). Bottom row: prior (black dashed line) and posterior (blue shading) distributions for $\mathcal{I}$ with true value (black vertical line).



output, $Y = \eta(\mathbf{X})$, is represented as a truncated polynomial chaos expansion (PCE):

$$Y = \eta(\mathbf{X}) \approx \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}), \tag{5.28}$$

where $a_{\boldsymbol{\alpha}}$ are expansion coefficients to be determined, and $\psi_{\boldsymbol{\alpha}}$ are known orthogonal multivariate polynomials which are chosen to correspond to the probability distribution $f_{\mathbf{X}}(\mathbf{x})$. The multivariate polynomials are constructed as the product of d univariate polynomials:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = \psi_{\alpha_1}(x_1) \times \cdots \times \psi_{\alpha_d}(x_d), \tag{5.29}$$

where the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ is an index defining the degree of each univariate polynomial.

The polynomials which appear in the truncated PCE of Equation (5.28) depend on the truncation scheme $|\boldsymbol{\alpha}|$. In this chapter, the tensor product truncation scheme (defined in Section 3.2.1) will be used following Section 3.2.2, where all combinations of univariate polynomials of degree 0 up to p are included. This results in the truncated PCE of Equation (5.28) having $N = (p+1)^d$ terms.

With the truncated PCE defined, an equation for each of the expansion coefficients can be derived using the orthogonality of the polynomials (Le Maître et al., 2002; Reagan et al., 2003). The expansion coefficients are given by:

$$a_{\boldsymbol{\alpha}} = \frac{1}{\gamma_{\boldsymbol{\alpha}}^2} \int_{\mathcal{X}} \psi_{\boldsymbol{\alpha}}(\mathbf{x}) \eta(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \qquad 0 \leq |\boldsymbol{\alpha}| \leq p, \tag{5.30}$$

where $\gamma_{\boldsymbol{\alpha}}^2$ are known constants related to the orthogonal polynomials. Since the solution of this integral is intractable, the NISP method solves the integral numerically using simulator runs $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(n)}\right)$ obtained at an experimental design $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$. Specifically, a tensor grid of one-dimensional Gaussian quadrature rules is employed to align with the choice of orthogonal polynomial basis. Here, the design size $n = (p+1)^d = N$, that is, exactly the number of terms in the truncated PCE.

The estimated coefficients, $\hat{a}_{\boldsymbol{\alpha}}$, $0 \leq |\boldsymbol{\alpha}| \leq p$, are substituted into the truncated PCE in Equation (5.28) to give the PC surrogate, which is used as a fast approximation to the expensive simulator:

$$Y = \eta(\mathbf{X}) \approx \hat{\eta}^{PC}(\mathbf{X}) = \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} \hat{a}_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}). \tag{5.31}$$

5.2.1. Application of Bayesian quadrature

The NISP method for PC uses numerical integration to estimate the coefficients of the expansion, introducing a source of error into the framework. This error is not accounted for, and the associated uncertainty may not be negligible since typically only a small number of evaluations can be made of the expensive integrand. This provides a good opportunity to apply the ideas of probabilistic numerics and Bayesian quadrature to this framework, to create a probabilistic interpretation of PC. This novel approach will be called probabilistic polynomial chaos (PPC), the methodology of which will be described in this section.

A number of assumptions and notational changes must be made to the NISP method for PC to comply with the BQ methodology outlined in Section 5.1.2. Firstly, the inputs are assumed to be independent and identically distributed as the standard Gaussian distribution as given by Equation (5.13). This in turn restricts the multivariate polynomials in the truncated PCE to be a product of d one-dimensional Hermite polynomials:

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) = H_{\alpha_1}(x_1) \times \cdots \times H_{\alpha_d}(x_d), \tag{5.32}$$

due to their orthogonality with respect to the Gaussian distribution in Equation (5.13). This means that the truncated PCE of Equation (5.28) is a Wiener-Hermite expansion (Ghanem and Spanos, 1991b).

For notational convenience, rewrite the truncated PCE of Equation (5.28) in vector form:

$$Y = \eta(\mathbf{X}) \approx \sum_{0 \leq |\boldsymbol{\alpha}| \leq p} a_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\mathbf{X}) \equiv \boldsymbol{\psi}(\mathbf{X})^\top \mathbf{a}, \tag{5.33}$$

where $\mathbf{a}$ and $\boldsymbol{\psi}(\mathbf{x})$ are $N \times 1$ vectors containing the coefficients $a_{\boldsymbol{\alpha}}$ and polynomials $\psi_{\boldsymbol{\alpha}}(\mathbf{x})$, $0 \leq |\boldsymbol{\alpha}| \leq p$, respectively. Then Equation (5.30) can be rewritten as:

$$\boldsymbol{\Gamma}\mathbf{a} = \int_{\mathcal{X}} \boldsymbol{\psi}(\mathbf{x}) \eta(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{5.34}$$

where $\boldsymbol{\Gamma}$ is a $N \times N$ diagonal matrix with diagonal elements $\gamma^2_{\boldsymbol{\alpha}}$, $0 \leq |\boldsymbol{\alpha}| \leq p$, that is, the known normalisation constants corresponding to the Hermite polynomial basis. Estimating the coefficients using the NISP method returns the vector $\hat{\mathbf{a}}$, which contains the estimated coefficients $\hat{a}_{\boldsymbol{\alpha}}$, $0 \leq |\boldsymbol{\alpha}| \leq p$. The PC surrogate, $\hat{\eta}^{PC}(\mathbf{x})$, is given by substituting the estimated coefficients, $\hat{\mathbf{a}}$, into the truncated PCE of Equation (5.33):

$$Y = \eta(\mathbf{X}) \approx \hat{\eta}^{PC}(\mathbf{X}) = \boldsymbol{\psi}(\mathbf{X})^\top \hat{\mathbf{a}}. \tag{5.35}$$

Note that Equation (5.34) is now in the form of the integral in Equation (5.2) in Section 5.1.2, with $\mathbf{r}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})$ (N known multivariate Hermite polynomials), $g(\mathbf{x}) = \eta(\mathbf{x})$ (an expensive black-box simulator) and $\mathcal{I} = \boldsymbol{\Gamma}\mathbf{a}$ (unknown expansion coefficients multiplied by known constants). It is now possible to apply the BQ methodology of Section 5.1.2 directly to the integral in Equation (5.34), to recast the NISP method for PC in a probabilistic framework.

Firstly, a standard GP prior is placed on the simulator $\eta(\mathbf{x})$, of the form:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(M(\mathbf{x}; \boldsymbol{\beta}), \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\right), \tag{5.36}$$

where the mean function is the regression model $M(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{h}^\top(\mathbf{x})\boldsymbol{\beta}$, and the correlation function is of the separable squared exponential form in Equation (5.5).

Secondly, n evaluations of the simulator, $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(n)}\right)$, are observed at an experimental design $\mathcal{D} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)$, and the posterior $\eta(\mathbf{x}) \,|\, \mathbf{y}$ is derived:

$$\eta(\mathbf{x}) \,|\, \mathbf{y} \sim t_{n-q}\left(M^{**}(\mathbf{x}), \hat{\sigma}^2 C^{**}(\mathbf{x}, \mathbf{x}')\right), \tag{5.37}$$

where expressions for the posterior mean and correlation functions, $M^{**}(\mathbf{x})$ and $C^{**}(\mathbf{x}, \mathbf{x}')$ respectively, as well as estimates for the hyperparameters $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$, are given in Chapter 3.

Next, the posterior for $\boldsymbol{\Gamma}\mathbf{a} \,|\, \mathbf{y}$ can be obtained using BQ, and is a location-scale shifted, multivariate Student's-t distribution with $n - q$ degrees of freedom. Its posterior mean vector and covariance matrix are:

$$\mathbb{E}\left[\boldsymbol{\Gamma}\mathbf{a} \,|\, \mathbf{y}\right] = \mathbf{R}\hat{\boldsymbol{\beta}} + \mathbf{T}\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\beta}}), \qquad n - q > 1, \tag{5.38}$$

$$\mathrm{var}\left[\boldsymbol{\Gamma}\mathbf{a} \,|\, \mathbf{y}\right] = \hat{\sigma}^2\mathbf{V}, \qquad n - q > 2, \tag{5.39}$$

with $\mathbf{V}$ defined in Equation (5.9). Again, note the restrictions on the degrees of freedom for the Student's-t distribution for its mean and variance to be defined. The expectation and variance, given by Equations (5.38) and (5.39) respectively, again depend on $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$. Since in PPC $\mathbf{r}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})$, these integrals are now

defined as:

$$\begin{aligned}\mathbf{R} &= \mathbb{E}\left[\boldsymbol{\psi}(\mathbf{x})\mathbf{h}(\mathbf{x})^\top\right] \\ &= \int_{\mathcal{X}} \boldsymbol{\psi}(\mathbf{x})\mathbf{h}(\mathbf{x})^\top f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x}, \qquad (5.40)\\ \mathbf{T} &= \mathbb{E}\left[\boldsymbol{\psi}(\mathbf{x})\mathbf{t}(\mathbf{x})^\top\right] \\ &= \int_{\mathcal{X}} \boldsymbol{\psi}(\mathbf{x})\mathbf{t}(\mathbf{x})^\top f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x}, \qquad (5.41)\\ \mathbf{U} &= \mathbb{E}\left[\mathbb{E}\left[C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})\boldsymbol{\psi}(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}')^\top\right]\right] \\ &= \int_{\mathcal{X}}\int_{\mathcal{X}} C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta})\boldsymbol{\psi}(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}')^\top f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}', \qquad (5.42)\end{aligned}$$

to be compared with Equations (5.10)–(5.12). Solutions to these integrals remain tractable and will be considered in Section 5.2.2.

Finally, the posterior distribution of the expansion coefficients given the simulator runs, $\mathbf{a}\,|\,\mathbf{y}$, is desired for use in the PC surrogate. This posterior will be denoted $\mathbf{a}^* = \mathbf{a}\,|\,\mathbf{y}$ for convenience, and can be obtained by modifying Equations (5.38) and (5.39) using the properties of expectation and variance. In particular, $\mathbf{a}^*$ is a location-scale shifted, multivariate Student's-t distribution with $n-q$ degrees of freedom, with posterior mean vector and covariance matrix:

$$\mathbb{E}\left[\mathbf{a}^*\right] = \mathbb{E}\left[\mathbf{a}\,|\,\mathbf{y}\right] = \boldsymbol{\Gamma}^{-1}\mathbb{E}\left[\boldsymbol{\Gamma}\mathbf{a}\,|\,\mathbf{y}\right], \qquad n-q>1, \qquad (5.43)$$

$$\begin{aligned}\mathrm{var}\left[\mathbf{a}^*\right] = \mathrm{var}\left[\mathbf{a}\,|\,\mathbf{y}\right] &= \boldsymbol{\Gamma}^{-1}\mathrm{var}\left[\boldsymbol{\Gamma}\mathbf{a}\,|\,\mathbf{y}\right](\boldsymbol{\Gamma}^\top)^{-1} \\ &= \boldsymbol{\Gamma}^{-1}\mathrm{var}\left[\boldsymbol{\Gamma}\mathbf{a}\,|\,\mathbf{y}\right]\boldsymbol{\Gamma}^{-1}, \qquad n-q>2, \qquad (5.44)\end{aligned}$$

since $\boldsymbol{\Gamma}$ is a diagonal matrix ($(\boldsymbol{\Gamma}^\top)^{-1} = \boldsymbol{\Gamma}^{-1}$), and where $\mathbb{E}\left[\boldsymbol{\Gamma}\mathbf{a}\,|\,\mathbf{y}\right]$ and $\mathrm{var}\left[\boldsymbol{\Gamma}\mathbf{a}\,|\,\mathbf{y}\right]$ are given by Equations (5.38) and (5.39) respectively. The application of BQ is now complete, with the posterior $\mathbf{a}^*$ quantifying the uncertainty in the PC coefficients given the finite evaluations of the simulator $\eta(\mathbf{x})$. The uncertain coefficients can be substituted into the truncated PCE in Equation (5.33) to give a probabilistic version of the standard PC surrogate in Equation (5.35). This will be demonstrated shortly in Section 5.2.3, after a number of issues regarding the implementation of the above methodology are discussed in the following section.

As a final note, placing a GP prior on the function $\eta(\mathbf{x})$ induces the following prior distribution for the expansion coefficients $\mathbf{a}$:

$$\mathbf{a}\,|\,\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta} \sim MVN(\boldsymbol{\Gamma}^{-1}\mathbf{R}\boldsymbol{\beta}, \sigma^2\boldsymbol{\Gamma}^{-1}\mathbf{U}\boldsymbol{\Gamma}^{-1}). \qquad (5.45)$$

5.2.2. Implementation

In this section, a number of issues affecting the implementation of the probabilistic PC methodology presented in Section 5.2.1 are discussed in the same vein as Section 5.1.3. Experimental design and analytical solutions to the **R**, **T** and **U** integrals are discussed. Further, the effect of multiplying the GP prior for $\eta(\mathbf{x})$ by a vector of multivariate polynomials $\boldsymbol{\psi}(\mathbf{x})$ is presented, and the traditional NISP method for PC is demonstrated to be a special case of probabilistic PC.

Properties of $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$

The integrand in Equation (5.34) is $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$, that is, an expensive simulator multiplied by an $N \times 1$ vector of known multivariate Hermite polynomials. The first step in BQ is to place a GP prior on $\eta(\mathbf{x})$, and this implicitly induces a prior on $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$. This prior has some interesting properties, which will now be examined.

Recall that the GP prior for $\eta(\mathbf{x})$ has the form:

$$\eta(\mathbf{x}) \,|\, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(M(\mathbf{x}; \boldsymbol{\beta}), V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta})\right), \tag{5.46}$$

where $M(\mathbf{x}; \boldsymbol{\beta}) \equiv \mathbb{E}\left[\eta(\mathbf{x})\right]$ denotes the mean function, dependent on hyperparameters $\boldsymbol{\beta}$, and $V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}) \equiv \mathrm{cov}\left[\eta(\mathbf{x}), \eta(\mathbf{x}')\right]$ denotes the covariance function, dependent on hyperparameters σ^2 and $\boldsymbol{\delta}$. A special case of the covariance is the GP variance, $\mathrm{var}\left[\eta(\mathbf{x})\right] \equiv \mathrm{cov}\left[\eta(\mathbf{x}), \eta(\mathbf{x})\right] = \sigma^2$.

The statistical properties of $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$ are given by modifying the mean and covariance functions in Equation (5.46). The expectation of $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$, $\mathbb{E}\left[\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})\right]$, is a $N \times 1$ vector with k-th element:

$$\begin{aligned} \mathbb{E}\left[\psi_k(\mathbf{x})\eta(\mathbf{x})\right] &= \psi_k(\mathbf{x})\,\mathbb{E}\left[\eta(\mathbf{x})\right] \\ &= \psi_k(\mathbf{x}) M(\mathbf{x}; \boldsymbol{\beta}), \end{aligned} \tag{5.47}$$

where $\psi_k(\mathbf{x})$ denotes the k-th element of the vector $\boldsymbol{\psi}(\mathbf{x})$. The variance of $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$, $\mathrm{var}\left[\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})\right]$, is an $N \times 1$ vector with k-th element:

$$\begin{aligned} \mathrm{var}\left[\psi_k(\mathbf{x})\eta(\mathbf{x})\right] &= \psi_k^2(\mathbf{x})\,\mathrm{var}\left[\eta(\mathbf{x})\right] \\ &= \psi_k^2(\mathbf{x})\sigma^2. \end{aligned} \tag{5.48}$$

More generally, the covariance of $\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x})$ at two input settings $\mathbf{x}$ and $\mathbf{x}'$,

$\text{cov}\left[\boldsymbol{\psi}(\mathbf{x})\eta(\mathbf{x}), \boldsymbol{\psi}(\mathbf{x}')\eta(\mathbf{x}')\right]$, is an $N \times N$ matrix with k, l-th element:

$$\begin{aligned}\text{cov}\left[\psi_k(\mathbf{x})\eta(\mathbf{x}), \psi_l(\mathbf{x}')\eta(\mathbf{x}')\right] &= \psi_k(\mathbf{x})\psi_l(\mathbf{x})\,\text{cov}\left[\eta(\mathbf{x}), \eta(\mathbf{x}')\right] \\ &= \psi_k(\mathbf{x})\psi_l(\mathbf{x})\,V(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}). \end{aligned} \tag{5.49}$$

Clearly, the form of the polynomials will have a large effect on the expectation, variance and covariance in Equations (5.47)–(5.49). In particular, $\mathbf{x}$ values which are roots or zeros of the polynomials, satisfying $\psi_k(\mathbf{x}) = 0$, correspond with zero variance, $\text{var}\left[\psi_k(\mathbf{x})\eta(\mathbf{x})\right] = 0$. This will be illustrated in the following example. Of course, the expressions in Equations (5.47)–(5.49) also hold for the posterior mean and covariance functions, $M^{**}(\mathbf{x})$ and $V^{**}(\mathbf{x}, \mathbf{x})$ respectively.

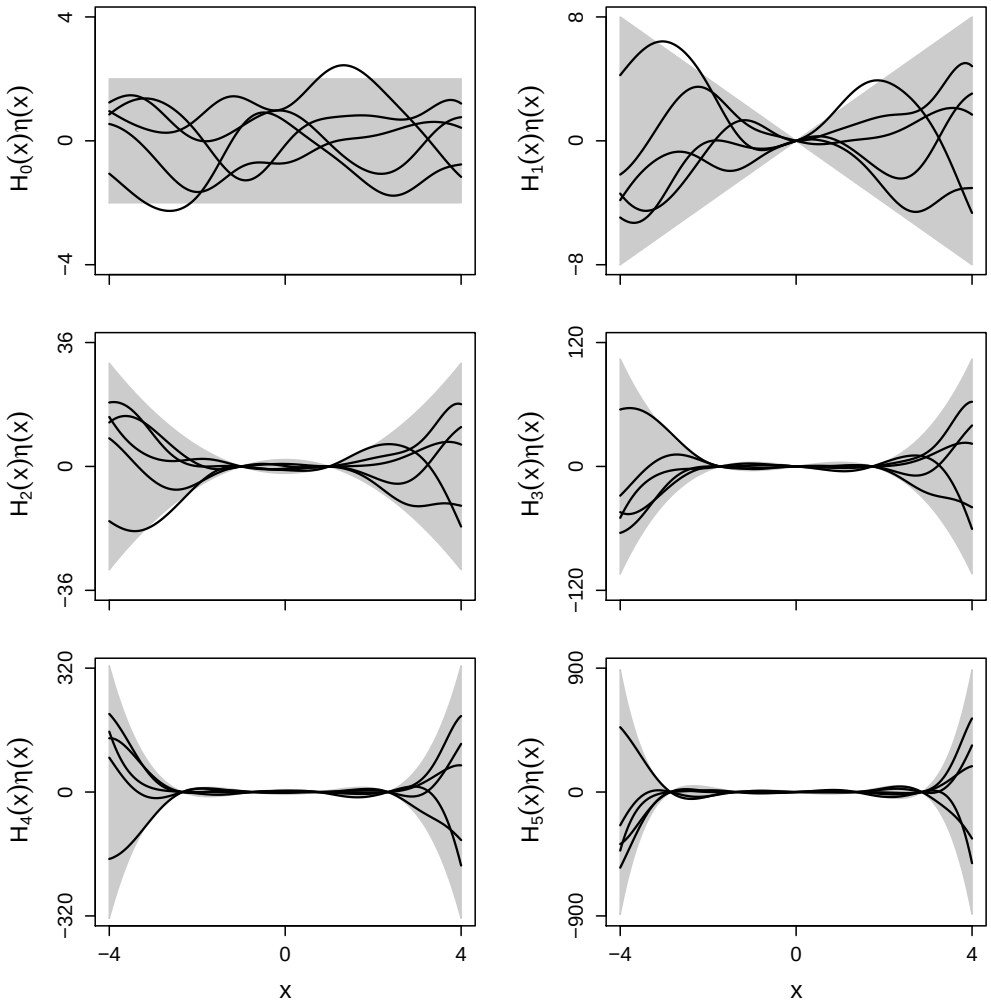
Consider a one-dimensional simulator, $\eta(x)$, $x \in [-4, 4]$, and the case where $\boldsymbol{\psi}(x)$ is a vector of univariate Hermite polynomials up to degree 5, $\boldsymbol{\psi}(x) = \{H_\alpha(x)\}_{\alpha=0}^{5}$, such that $N = 6$. The univariate Hermite polynomials are given in Example 3.2.2 in Chapter 3. Place a GP prior on $\eta(x)$ of the form in Equation (5.46). Assume without loss of generality that $M(x; \boldsymbol{\beta}) = 0$ (zero mean) and use the squared exponential covariance function $V(x, x'; \sigma^2, \delta) = \sigma^2 \exp(-(x - x')^2/2\delta^2)$, with $\sigma^2 = 1$ and $\delta = 1$. Figure 5.3 shows five draws from this GP prior modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$. Prior uncertainty is represented as a region of two standard deviations around the prior mean, $0 \pm 2\sqrt{\text{var}\left[H_\alpha(x)\eta(x)\right]} = 0 \pm 2\sqrt{H_\alpha^2(x)\sigma^2} = 0 \pm 2H_\alpha(x)\sigma$, and plotted across the input range $x \in [-4, 4]$ in each case.

Each draw from the prior for $H_\alpha(x)\eta(x)$, $\alpha = 0, \ldots, 5$, varies around its expectation $\mathbb{E}\left[H_\alpha(x)\eta(x)\right] = H_\alpha(x)M(x; \boldsymbol{\beta}) = 0$. The Hermite polynomials vary on a larger scale as the polynomial degree α is increased, and the draws from the $H_\alpha(x)\eta(x)$ priors respect this (note the change in the y-axes scales). Notably, $H_0(x) = 1$, so the draws from the prior for $H_0(x)\eta(x)$ are simply draws from the original GP prior. In terms of the variance, each Hermite polynomial $H_\alpha(x)$ has α zeros, so correspondingly the draws from $H_\alpha(x)\eta(x)$ have α points where the prior variance is zero. There is also covariance between $H_k(x)\eta(x)$ and $H_l(x)\eta(x)$, for $k, l = 0, \ldots, 5$ and $k \neq l$, given by Equation (5.49), but this is not shown here. Variation of the σ^2 and δ hyperparameters would give rise to similar behaviour as shown in Figure 3.6.

Analytical solutions of R, T and U

Consider again the solutions of the **R**, **T** and **U** integrals in Equations (5.10)–(5.12). In Section 5.1.3 it was shown that these integrals have analytical solutions under a number of assumptions. In particular, the GP correlation function $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$ is assumed to be of the separable squared exponential form in Equation (5.5). Moreover, the probability distribution $f_{\mathbf{X}}(\mathbf{x})$ is assumed to be the product of d independent

Figure 5.3.: Five draws from one-dimensional Gaussian process priors (black) modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$. The mean function is a constant $\beta_1 = 0$ and the covariance function is squared exponential with variance $\sigma^2 = 1$ and correlation length $\delta = 1$. Prior uncertainty is represented as two standard deviations around the prior mean, $0 \pm 2\sqrt{\text{var}\left[H_\alpha(x)\eta(x)\right]}$ (grey shading), in each case.



standard Gaussian distributions as given by Equation (5.13), and the k-th elements of the $\mathbf{h}(\mathbf{x})^\top$ and $\mathbf{r}(\mathbf{x})$ vectors are assumed to separate into the product of d terms, shown in Equations (5.15) and (5.16) respectively. Under these assumptions of separability, the elements of the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ matrices are made up of the product of d univariate integrals, defined as $R_{k,l}^{(j)}$, $T_{k,l}^{(j)}$ and $U_{k,l}^{(j)}$, $j = 1, \ldots, d$, in Equations (5.17)–(5.19) respectively.

In Section 5.1.3 it was then assumed, without loss of generality, that $\mathbf{r}(\mathbf{x}) = (1)$, such that the solutions of $R_l^{(j)}$, $T_l^{(j)}$ and $U^{(j)}$, given in Equations (5.20)–(5.22) respectively, are required. Solutions for these simplified integrals were then given in Equations (5.23)–(5.25) using standard results from the BQ literature (O'Hagan, 1991; Rasmussen and Ghahramani, 2002). However, for the PPC methodology presented in Section 5.2.1, $\mathbf{r}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})$, that is, a vector of $q_r = N$ known multivariate polynomials, $\psi_{\boldsymbol{\alpha}}(\mathbf{x})$, $0 \leq |\boldsymbol{\alpha}| \leq p$. As a consequence, $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ in the form of Equations (5.40)–(5.42) are of interest, and solutions to the more general integrals $R_{k,l}^{(j)}$, $T_{k,l}^{(j)}$ and $U_{k,l}^{(j)}$, $j = 1, \ldots, d$, are required.

The multivariate polynomials in the vector $\boldsymbol{\psi}(\mathbf{x})$ are made up of the product of d univariate Hermite polynomials, as given in Equation (5.32). Define the k-th element of $\boldsymbol{\psi}(\mathbf{x})$ as $\psi_k(\mathbf{x}) = \psi_{\boldsymbol{\alpha}_k}(\mathbf{x})$, where:

$$\psi_{\boldsymbol{\alpha}_k}(\mathbf{x}) = H_{\alpha_{k1}}(x_1) \times \cdots \times H_{\alpha_{kd}}(x_d),$$

where the vector $\boldsymbol{\alpha}_k = (\alpha_{k1}, \ldots, \alpha_{kd}) \in \mathbb{N}_0^d$ is an index defining the polynomial degree of each univariate polynomial in the k-th multivariate polynomial of $\boldsymbol{\psi}(\mathbf{x})$. With this notation, it can be noted that $r_k(x_j) = H_{\alpha_{kj}}(x_j)$, namely, the Hermite polynomial of degree α_{kj} in the input x_j. The definition of $r_k(x_j)$ was given in Equation (5.15). Hence, in PPC $R_{k,l}^{(j)}$, $T_{k,l}^{(j)}$ and $U_{k,l}^{(j)}$ take the following form:

$$R_{k,l}^{(j)} = \int_{-\infty}^{\infty} H_{\alpha_{kj}}(x_j) h_l(x_j) \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j, \qquad (5.50)$$
$$k = 1, \ldots, N,\ l = 1, \ldots, q,$$

$$T_{k,l}^{(j)} = \int_{-\infty}^{\infty} H_{\alpha_{kj}}(x_j) \exp\left[-\frac{1}{2}\left(\frac{x_j - x_j^{(l)}}{\delta_j}\right)^2\right] \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j, \qquad (5.51)$$
$$k = 1, \ldots, N,\ l = 1, \ldots, n,$$

$$U_{k,l}^{(j)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{x_j - x_j'}{\delta_j}\right)^2\right] H_{\alpha_{kj}}(x_j) H_{\alpha_{lj}}(x_j') \frac{e^{-x_j^2/2}}{\sqrt{2\pi}} \frac{e^{-x_j'^2/2}}{\sqrt{2\pi}} \, \mathrm{d}x_j \, \mathrm{d}x_j', \qquad (5.52)$$
$$k = 1, \ldots, N,\ l = 1, \ldots, N.$$

Like standard BQ, for the PPC methodology to work it is required that these inte-

grals have analytical solutions, and these will now be presented. Firstly, the solution of $R_{k,l}^{(j)}$ in Equation (5.50) will be discussed. Note that $R_{k,l}^{(j)} = \mathbb{E}\left[H_{\alpha_{kj}}(X_j)h_l(X_j)\right]$, that is, the expectation of the function $H_{\alpha_{kj}}(X_j)h_l(X_j)$ taken with respect to the standard Gaussian distribution. Recall that $h_l(X_j)$ is the basis function for the input X_j in the l-th element of the GP mean function, given by Equation (5.16). Two forms for the basis function $h_l(X_j)$ will be assumed here. Firstly, assume that this function is of the form $h_l(X_j) = X_j^b$, $b \in \mathbb{N}_0$, as was the case in Section 5.1.3. In this case, $R_{k,l}^{(j)} = \mathbb{E}\left[H_{\alpha_{kj}}(X_j)X_j^b\right]$, that is, the expectation of the univariate Hermite polynomial of degree α_{kj} in input X_j multiplied by X_j^b. This can be rewritten as:

$$R_{k,l}^{(j)} = \mathbb{E}\left[c_{\alpha_{kj}+b}X_j^{\alpha_{kj}+b} + \ldots + c_1X_j + c_0\right]$$

that is, the expectation of a polynomial of order $\alpha_{kj} + b$ with known coefficients $c_i \in \mathbb{Z}$, $i = 1, \ldots, \alpha_{kj} + b$. Employing the linearity of expectation gives:

$$R_{k,l}^{(j)} = c_{\alpha_{kj}+b}\mathbb{E}\left[X_j^{\alpha_{kj}+b}\right] + \ldots + c_1\mathbb{E}\left[X_j\right] + c_0, \tag{5.53}$$

namely, a linear function of moments of the standard Gaussian distribution, for which Equation (5.23) in Section 5.1.3 can be applied. In this case, the solution to $R_{k,l}^{(j)}$ is a linear combination of double factorial numbers. Secondly, consider the interesting case when the basis function is itself a Hermite polynomial, $h_l(X_j) = H_{\alpha_{lj}}(X_j)$. When this is the case, the vector of basis functions $\mathbf{h}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})$, $q = N$, and the GP mean function takes the form:

$$M(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\beta}, \tag{5.54}$$

which is equivalent to the truncated PCE in Equation (5.33). As will be discussed in the next section, with this choice of mean function the NISP method for PC can be recovered as a special case of PPC, so is the recommended choice. Here, $R_{k,l}^{(j)} = \mathbb{E}\left[H_{\alpha_{kj}}(X_j)H_{\alpha_{lj}}(X_j)\right]$, and the orthogonality of the Hermite polynomials can be employed (refer to Equation (3.6) in Chapter 3):

$$R_{k,l}^{(j)} = \int_{-\infty}^{\infty} H_{\alpha_{kj}}(x_j)H_{\alpha_{lj}}(x_j)\frac{e^{-x_j^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x_j = \alpha_{kj}!\delta_{\alpha_{kj}\alpha_{lj}}, \tag{5.55}$$

In this case $\mathbf{R} = \boldsymbol{\Gamma}$, the $N \times N$ diagonal matrix of known normalisation constants for the Hermite polynomials defined in Equation (5.34).

Obtaining the solutions of $T_{k,l}^{(j)}$ and $U_{k,l}^{(j)}$ in Equations (5.51) and (5.52) involves the repeated application of integration by parts (since $H_{\alpha_{kj}}(x_j)$ is just a polynomial), and the results for $T_l^{(j)}$ and $U^{(j)}$ given in Equations (5.24) and (5.25). The solution

Table 5.1.: The construction of the modified Hermite polynomials, $H^*_\alpha(x,y)$, $\alpha = 0,\ldots,5$, using the standard Hermite polynomials, $H_\alpha(x)$, and the polynomial $\sum_{i=0}^{|H_\alpha(x)|-1} y^i$, where $|H_\alpha(x)|$ denotes the number of terms in $H_\alpha(x)$.

α	$H_\alpha(x)$	$\sum_{i=0}^{\lvert H_\alpha(x)\rvert-1} y^i$	$H^*_\alpha(x,y)$
0	1	1	1
1	x	1	x
2	x^2-1	$1+y$	x^2-y
3	x^3-3x	$1+y$	x^3-3xy
4	x^4-6x^2+3	$1+y+y^2$	$x^4-6x^2y+3y^2$
5	x^5-10x^3+15x	$1+y+y^2$	$x^5-10x^3y+15xy^2$

of $T^{(j)}_{k,l}$ is:

$$T^{(j)}_{k,l} = \exp\left[\frac{-x_j^{(l)2}}{2(\delta_j^2+1)}\right] \frac{H^*_{\alpha_{kj}}\left(x_j^{(l)}, (\delta_j^2+1)\right)}{(\delta_j^2+1)^{\alpha_{kj}}\sqrt{1+\frac{1}{\delta_j^2}}}, \tag{5.56}$$

for $j = 1,\ldots,d$, $k = 1,\ldots,N$ and $l = 1,\ldots,n$. Here, $H^*_\alpha(x,y)$ defines a modified Hermite polynomial of degree α in terms of parameters x and y. Specifically, $H^*_\alpha(x,y)$ is constructed as the Hermite polynomial of degree α in x, $H_\alpha(x)$, combined with a polynomial in y of the form $\sum_{i=0}^{|H_\alpha(x)|-1} y^i$, where $|H_\alpha(x)|$ denotes the number of terms in $H_\alpha(x)$. The construction of the modified Hermite polynomials, $H^*_\alpha(x,y)$, from $H_\alpha(x)$ and $\sum_{i=0}^{|H_\alpha(x)|-1} y^i$, is shown in Table 5.1, for $\alpha = 0,\ldots,5$.

Finally, the solution of $U^{(j)}_{k,l}$ is:

$$U^{(j)}_{k,l} = \begin{cases} (-1)^{(\alpha_{kj}-\alpha_{lj})/2} \frac{DF((\alpha_{kj}+\alpha_{lj})/2)}{(\delta_j^2+2)^{(\alpha_{kj}+\alpha_{lj})/2}\sqrt{1+\frac{2}{\delta_j^2}}} & \alpha_{kj}+\alpha_{lj} \text{ even}, \\ 0 & \alpha_{kj}+\alpha_{lj} \text{ odd}, \end{cases} \tag{5.57}$$

for $j = 1,\ldots,d$ and $k,l = 1,\ldots,N$. The sequence $DF(n)$ was defined as the double factorial of odd numbers in Section 5.1.3.

Recovering traditional polynomial chaos

Recall that for traditional PC using the NISP method to estimate the coefficients, a tensor grid of one-dimensional Gaussian quadrature rules is used for the experimental design. Specifically for the case of a Wiener-Hermite expansion, one-dimensional Gauss-Hermite quadrature rules are used to align with the Gaussian probability distribution for the inputs, which appears in the NISP integrals for the coefficients. The size of this tensor grid design is $n = (p+1)^d$ since one-dimensional quadrature rules of size $p+1$ are used in each of the d input directions, to fit a global polynomial function of degree p. This is naturally equal to the number of terms in the truncated PCE, N, due to the use of a tensor product truncation scheme: $n = N = (p+1)^d$.

It has been shown by various authors (Diaconis, 1988; Minka, 2000; Särkkä et al., 2016) that classical quadrature rules can be recast in a probabilistic framework with certain choices for the GP correlation function and input probability distribution. In particular, the squared exponential correlation function and Gaussian probability distribution lead to a probabilistic version of the Gauss-Hermite quadrature rule (Diaconis, 1988). These are the exact choices for the correlation function and distribution in the PPC methodology presented in Section 5.2.1. Therefore, PPC is indeed a probabilistic version of the NISP method for PC.

Exact results from the traditional NISP method for PC can be recovered as a special case of the probabilistic PC methodology, as follows. Firstly, the experimental design must be equivalent to that used for the traditional PC approach — a tensor grid of one-dimensional Gauss-Hermite quadrature rules. Secondly, the GP mean function must be of the form $M(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\beta}$, that is, the truncated PCE of order p. The use of Hermite polynomials as basis functions means that the quadrature weights derived by BQ in Equation (5.26) are identical to the weights given by the Gauss-Hermite quadrature rule (O'Hagan, 1991). The number of mean function terms, q, is now equal to the number of terms in the truncated PCE, N, which in turn is equal to the number of design points, n: $q = N = n$. As a consquence, the matrix $\mathbf{H}$ is square and nonsingular, and the derivation of the GP posterior $\eta(\mathbf{x}) \,|\, \mathbf{y}$ reduces to a simple form. Notably, $\mathbf{H}^\top \mathbf{A}^{-1} = \mathbf{I}$, giving $\hat{\boldsymbol{\beta}} = \mathbf{H}^{-1}\mathbf{y}$, and the GP posterior mean function becomes the fitted regression model, $M^{**}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^\top \hat{\boldsymbol{\beta}}$ (O'Hagan, 1991), which in this case is a PC surrogate itself (it will shortly be proved that this is exactly the PC surrogate given by the NISP method). When this GP posterior is projected down to a posterior distribution for the PC coefficients $\mathbf{a}^*$ using BQ, the mean of the posterior is:

$$\begin{aligned}
\mathbb{E}\left[\mathbf{a}^*\right] = \mathbb{E}\left[\mathbf{a} \,|\, \mathbf{y}\right] &= \boldsymbol{\Gamma}^{-1}\mathbb{E}\left[\boldsymbol{\Gamma}\mathbf{a} \,|\, \mathbf{y}\right] \\
&= \boldsymbol{\Gamma}^{-1}(\mathbf{R}\hat{\boldsymbol{\beta}} + \mathbf{T}\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\beta}})) \\
&= \boldsymbol{\Gamma}^{-1}(\mathbf{R}\mathbf{H}^{-1}\mathbf{y} + \mathbf{T}\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\mathbf{H}^{-1}\mathbf{y})) \\
&= \boldsymbol{\Gamma}^{-1}(\mathbf{R}\mathbf{H}^{-1}\mathbf{y} + \mathbf{T}\mathbf{A}^{-1}(\mathbf{y} - \mathbf{y})) \\
&= \boldsymbol{\Gamma}^{-1}\mathbf{R}\mathbf{H}^{-1}\mathbf{y}, \qquad (5.58)
\end{aligned}$$

using Equations (5.38) and (5.43), as well as $\hat{\boldsymbol{\beta}} = \mathbf{H}^{-1}\mathbf{y}$. The BQ weights from

Equation (5.26) can also be shown to be:

$$\begin{aligned}\mathbf{w} &= \mathbf{T}\mathbf{A}^{-1} + (\mathbf{R} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^\top\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^\top\mathbf{A}^{-1} \\ &= \mathbf{T}\mathbf{A}^{-1} + (\mathbf{R} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H})\mathbf{H}^{-1} \\ &= \mathbf{T}\mathbf{A}^{-1} + \mathbf{R}\mathbf{H}^{-1} - \mathbf{T}\mathbf{A}^{-1}\mathbf{H}\mathbf{H}^{-1} \\ &= \mathbf{T}\mathbf{A}^{-1} + \mathbf{R}\mathbf{H}^{-1} - \mathbf{T}\mathbf{A}^{-1} \\ &= \mathbf{R}\mathbf{H}^{-1}, \end{aligned} \tag{5.59}$$

using $\mathbf{H}^\top\mathbf{A}^{-1} = \mathbf{I}$. The result of Equation (5.59) means that Equation (5.58) can be written as the quadrature rule $\mathbb{E}[\mathbf{a}^*] = \boldsymbol{\Gamma}^{-1}(\mathbf{w}\cdot\mathbf{y})$. Since the weights, $\mathbf{w}$, are exactly the Gauss-Hermite weights (due to choices of the mean and covariance functions) and $\mathbf{y}$ is obtained by evaluating the simulator at Gauss-Hermite quadrature points, the BQ posterior mean coincides exactly with the quadrature rule from the NISP method for PC. Therefore the BQ mean is exactly the PC coefficients estimated using NISP: $\mathbb{E}[\mathbf{a}^*] = \hat{\mathbf{a}}$.

In the previous section, it was also noted that $\mathbf{R} = \boldsymbol{\Gamma}$ for this choice of mean function. This means that Equation (5.58) can also be written as:

$$\begin{aligned}\mathbb{E}[\mathbf{a}^*] = \mathbb{E}[\mathbf{a}\,|\,\mathbf{y}] &= \boldsymbol{\Gamma}^{-1}\mathbf{R}\mathbf{H}^{-1}\mathbf{y} \\ &= \boldsymbol{\Gamma}^{-1}\boldsymbol{\Gamma}\mathbf{H}^{-1}\mathbf{y} \\ &= \mathbf{H}^{-1}\mathbf{y} = \hat{\boldsymbol{\beta}}.\end{aligned}$$

Hence, the estimated GP mean function hyperparameters are exactly the PC coefficients estimated using the traditional NISP method: $\hat{\boldsymbol{\beta}} = \hat{\mathbf{a}}$. As a consequence, the GP posterior mean function is exactly the PC surrogate in Equation (5.35), $M^{**}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^\top\hat{\boldsymbol{\beta}} = \boldsymbol{\psi}(\mathbf{x})^\top\hat{\mathbf{a}} = \hat{\eta}^{PC}(\mathbf{x})$.

However, the fact that the number of design points, n, is equal to the number of mean function terms q, means that no information can be provided in the estimation of σ^2 and $\boldsymbol{\delta}$ when deriving the GP posterior $\eta(\mathbf{x})\,|\,\mathbf{y}$ (recall that $n \geq q + d + 1$ to be able to estimate the GP hyperparameters). In turn, the BQ posterior for the coefficients has $n - q = 0$ degrees of freedom and is therefore improper. As stated in Equations (5.43) and (5.44), the design size must satisfy $n > q + 2$ for the BQ posterior mean vector and covariance matrix to be defined. While it has been shown that the NISP method for PC can be recovered as a special case of the PPC methodology, no uncertainty information can be provided due to the restrictions on the design size. A direct interpretation of the NISP method for PC is therefore of little use as a probabilistic numerical method, as the interest here is in quantifying the uncertainty in the value of the integral (the PC coefficients) given a finite number of evaluations of the integrand (the expensive simulator). Following on from the

discussion on experimental design for BQ given in Section 5.1.3, the interest in PPC is the derivation of novel quadrature rules for the NISP method for PC by virtue of using different design strategies. The novel quadrature rules may also lead to an improved performance over traditional PC. Furthermore, the GP prior mean function does not have to be of the form $M(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\beta}$, where the number of basis functions can potentially be very large, $q = (p+1)^d$. Fitting a simpler prior mean function, such as the constant mean $M(\mathbf{x}; \boldsymbol{\beta}) = \beta$, vastly reduces q and may allow for smaller design sizes than in traditional PC. This also places more focus on the estimation of the important variance and correlation hyperparameters.

Experimental design

Similarly to standard BQ, the restriction on the experimental design for the PPC method is minimal, provided that the design size is large enough. As discussed in the previous section, the design size must satisfy $n > q + 2$ for the posterior mean and variance to be defined in the posterior for the PC coefficients. Furthermore, $n \geq q + d + 1$ to be able to estimate the GP hyperparameters. This rules out the design strategy for the traditional NISP method for PC, where $n = q$. Hence, the PPC method requires at least $d + 1$ more design points than traditional PC, if the GP prior mean function is a truncated PCE itself. This is natural considering the extra hyperparameters which need to be estimated in deriving the GP posterior, and a small price to pay for the valuable uncertainty information which the method provides. However, as already mentioned, the flexibility of the GP allows for simpler mean functions, such as the constant mean where $q = 1$. In this case, the design size must satisfy $n > d + 2$, potentially smaller than that for traditional PC.

Provided that the requirements on design size are satisfied, for PPC the design points can be located anywhere in the input domain $\mathcal{X}$, giving a more flexible approach than the NISP method for PC. Efficient experimental design for PPC is an interesting and open question, and a number of plausible design strategies could be considered. It is straightforward to apply popular techniques in probabilistic integration, such as BMC, BQMC and BMCMC, to PPC. Theoretical guarantees of convergence for these approaches are assumed to carry across to PPC, but have not been studied in this work. Alternatively, even though the traditional experimental design in the NISP method for PC does not satisfy the restrictions on design size, it could be combined with an additional design so that the variance components of the GP emulator can be estimated. For example, a tensor grid of one-dimensional Gauss-Hermite quadrature rules could be combined with another design — such as a Latin Hypercube or Sobol sequence — which is optimised for estimating the GP hyperparameters. As mentioned in Chapter 3, these latter designs are popular in the GP emulation literature, so this strategy would be a nice combination of typical PC

and GP experimental designs. However, it is not obvious exactly how this should be done and it remains a topic for future research.

5.2.3. Probabilistic polynomial chaos surrogate

The probabilistic polynomial chaos surrogate, denoted $\hat{\eta}^{PPC}(\cdot)$, is given by substituting the posterior distribution for the PC coefficients, $\mathbf{a}^*$, into the truncated PCE of Equation (5.33):

$$Y = \eta(\mathbf{X}) \approx \hat{\eta}^{PPC}(\mathbf{X}) \equiv \boldsymbol{\psi}(\mathbf{X})^\top \mathbf{a}^*. \tag{5.60}$$

It is a fully probabilistic surrogate for the simulator, with mean function:

$$\begin{aligned} \mathbb{E}\left[\hat{\eta}^{PPC}(\mathbf{x})\right] &= \mathbb{E}\left[\boldsymbol{\psi}(\mathbf{x})^\top \mathbf{a}^*\right] \\ &= \boldsymbol{\psi}(\mathbf{x})^\top \mathbb{E}\left[\mathbf{a}^*\right], \end{aligned} \tag{5.61}$$

and covariance function:

$$\begin{aligned} \operatorname{cov}\left[\hat{\eta}^{PPC}(\mathbf{x}), \hat{\eta}^{PPC}(\mathbf{x}')\right] &= \operatorname{cov}\left[\boldsymbol{\psi}(\mathbf{x})^\top \mathbf{a}^*, \boldsymbol{\psi}(\mathbf{x}')^\top \mathbf{a}^*\right] \\ &= \boldsymbol{\psi}(\mathbf{x})^\top \operatorname{var}\left[\mathbf{a}^*\right] \boldsymbol{\psi}(\mathbf{x}'), \end{aligned} \tag{5.62}$$

where $\mathbb{E}\left[\mathbf{a}^*\right]$ and $\operatorname{var}\left[\mathbf{a}^*\right]$ are defined in Equations (5.43) and (5.44) respectively. Since the surrogate is simply a polynomial function with uncertain coefficients, it can be evaluated very cheaply. Consequently, the simulator output at any new input setting $\mathbf{x}^{(*)}$ can be easily predicted as $\mathbb{E}\left[\hat{\eta}^{PPC}(\mathbf{x}^{(*)})\right] = \boldsymbol{\psi}(\mathbf{x}^{(*)})^\top \mathbb{E}\left[\mathbf{a}^*\right]$, with uncertainty in the prediction quantified as $\operatorname{var}\left[\hat{\eta}^{PPC}(\mathbf{x}^{(*)})\right] = \boldsymbol{\psi}(\mathbf{x}^{(*)})^\top \operatorname{var}\left[\mathbf{a}^*\right] \boldsymbol{\psi}(\mathbf{x}^{(*)})$. This enables fast MC (or a more efficient quadrature method) estimation of any function, $g(\cdot)$, of the uncertain simulator output, Y, as:

$$\mathbb{E}\left[g(Y)\right] \approx \frac{1}{m} \sum_{i=1}^{m} g(\hat{\eta}^{PPC}(\mathbf{x}^{(i)})),$$

for a suitably large sample of size m from the input distribution $f_{\mathbf{X}}(\mathbf{x})$. In performing such a MC estimation, the mean function in Equation (5.61) may be used to give a point estimate. However, because the PPC surrogate is probabilistic, any summary of the simulator output is itself a random variable. Hence, the posterior covariance function in Equation (5.62), or a sample from the posterior in Equation (5.60), can be used to quantify the uncertainty. In this way, the PPC surrogate can be used to efficiently tackle the UQ objectives outlined in Section 2.2.1 in Chapter 2.

Statistical moments

Recall from Chapter 3 that for traditional PC, estimates of the simulator output mean and variance are available directly from the estimated PC coefficients using Equations (3.23) and (3.24) respectively. In the PPC framework, the PC coefficients have a joint posterior distribution, so probabilistic estimates of the simulator output mean and variance may be obtained.

The mean of the uncertain simulator output, $\mathbb{E}[Y]$, is estimated as the first (zero order) coefficient of the PCE, as given by Equation (3.23). Using PPC, the mean of the uncertain simulator output given the observed simulator runs, $\mathbb{E}[Y \mid \mathbf{y}]$, is distributed as the first element of the $\mathbf{a}^*$ vector (corresponding to the zero order coefficient). Denote the first element of the vector $\mathbf{a}^*$ as $\mathbf{a}_1^*$. This is distributed as a location-scale shifted Student's-t distribution with $n - q$ degrees of freedom with posterior mean and variance:

$$\mathbb{E}[\mathbf{a}_1^*] = \mathbb{E}[\mathbf{a}^*]_1, \qquad n - q > 1, \tag{5.63}$$

$$\operatorname{var}[\mathbf{a}_1^*] = \operatorname{var}[\mathbf{a}^*]_{1,1}, \qquad n - q > 2, \tag{5.64}$$

that is, the first element of the posterior mean vector $\mathbb{E}[\mathbf{a}^*]$ and the first diagonal term of the posterior covariance matrix $\operatorname{var}[\mathbf{a}^*]$, given in Equations (5.43) and (5.44) respectively.

The variance of the uncertain simulator output, $\operatorname{var}[Y]$, is estimated as the sum in Equation (3.24). Importantly, the zero order term is not included in the sum. Using PPC, the variance of the uncertain simulator output given the observed simulator runs, $\operatorname{var}[Y \mid \mathbf{y}]$, is distributed as the posterior:

$$\operatorname{var}[Y \mid \mathbf{y}] \sim \mathbf{a}_{-1}^{*\top} \boldsymbol{\Gamma}_{\{-1,-1\}} \mathbf{a}_{-1}^*, \tag{5.65}$$

where $\mathbf{a}_{-1}^*$ denotes the posterior of the coefficients excluding the first (zero order) term, and $\boldsymbol{\Gamma}_{\{-1,-1\}}$ denotes the $\boldsymbol{\Gamma}$ matrix of normalisation constants with the first row and column removed. The analytical form of this posterior is non-trivial to derive, but samples from it can be easily and quickly obtained by sampling from the posterior for $\mathbf{a}^*$, meaning that the uncertainty on the variance of the simulator output may be obtained empirically using MC estimation.

5.2.4. Examples

In this section, PPC surrogates are applied to a simple one-dimensional function. Consider the following simulator in a single input x:

$$y = \eta(x) = \exp(-\sin^2(3x) - x^2). \tag{5.66}$$

This is the same function as used in Section 5.1.4 and in Hennig et al. (2015). Suppose that the single input is distributed as a standard Gaussian, $X \sim N(0, 1)$, and consider building a PPC surrogate for the uncertain simulator output $Y = \eta(X)$. Firstly, as in traditional PC, the simulator output is represented as a truncated PCE of order p in a Hermite polynomial basis:

$$Y \approx \sum_{\alpha=0}^{p} a_\alpha H_\alpha(X) = \boldsymbol{\psi}(x)^\top \mathbf{a}, \tag{5.67}$$

where $\boldsymbol{\psi}(x)$ and $\mathbf{a}$ are vectors containing the Hermite polynomials, $H_\alpha(x)$, and PC coefficients, a_α, $\alpha = 0, \ldots, 5$. The truncated PCE has $N = p + 1$ terms. For the purposes of this example, fix $p = 5$ so that $N = 6$. The coefficients are found by solving the integral:

$$\boldsymbol{\Gamma}\mathbf{a} = \int_{\mathcal{X}} \boldsymbol{\psi}(x)\eta(x)\frac{e^{-x^2/2}}{\sqrt{2\pi}}\,\mathrm{d}x \tag{5.68}$$

where $\boldsymbol{\Gamma}$ is an $N \times N$ diagonal matrix containing the diagonal elements $\alpha!$, $\alpha = 0, \ldots, 5$, namely, the known normalisation constants for the Hermite polynomials in $\boldsymbol{\psi}(x)$. In traditional PC, a Gauss-Hermite quadrature rule of size $n = N = 6$ is used to estimate the coefficients. The estimated coefficients, $\hat{\mathbf{a}}$, are then substituted into the truncated PCE to form the PC surrogate, $\hat{\eta}^{PC}(x) = \boldsymbol{\psi}(x)^\top \hat{\mathbf{a}}$.

Instead, the PPC methodology outlined in Sections 5.2.1 and 5.2.2 is used to find a posterior for the coefficients given a set of simulator runs, and hence a probabilistic PC surrogate. There are two main choices for the user in building a PPC surrogate: firstly, the specification of the GP prior mean function $M(x; \boldsymbol{\beta})$ (since the covariance function is assumed to be squared exponential), and secondly, the size and type of the experimental design $\mathcal{D}$ used to derive the posteriors. For the prior mean function, two choices will be featured in this example. Firstly, the constant prior mean $M(x; \boldsymbol{\beta}) = \beta$, such that $q = 1$, and secondly, the case where the prior mean function is the truncated PCE itself, $M(x; \boldsymbol{\beta}) = \boldsymbol{\psi}(x)^\top \boldsymbol{\beta}$, such that $q = N = 6$. For the experimental design, two strategies will be used. Firstly, a QMC sample from a Sobol sequence of size n is used, resulting in a BQMC approach to PPC. Requiring $n > q + 2$ to obtain variance information, and $n \geq q + d + 1$ to estimate the hyperparameters of the GP emulator, two design sizes are implemented: $n = 10$ and $n = 15$. Secondly, the Gauss-Hermite quadrature rule design from traditional

PC is combined with a small QMC sample which is suited to estimating the variance hyperparameters of the GP. Specifically, the Gauss-Hermite design of size 6 is combined with QMC samples from a Sobol sequence of size 4 and 9 to create designs of total size $n = 10$ and $n = 15$, aligning with the first design strategy. This second design strategy is combined with the case of a truncated PCE for the prior mean function — giving a direct interpretation of traditional PC, with additional design points allowing variance information to be obtained. The case of a QMC design is combined with the simple constant prior mean function to see if the results are noticeably different.

Results from building a PPC surrogate using the QMC sample of size $n = 10$ are shown in Figures 5.4–5.7. Additional results from the other design strategies are presented in Appendix A. Figure 5.4 shows the GP posterior fit to the simulator $\eta(x)$, modified by each of the Hermite polynomials, $H_\alpha(x)$, $\alpha = 0, \ldots, 5$. These functions are the integrands in the NISP method for PC, for which BQ is applied. In each case, the GP posterior mean and covariance functions are modified by the Hermite polynomials using Equations (5.47)–(5.49) in Section 5.2.2. This gives rise to regions in the input domain where the posterior variance is zero despite having not observed any simulator runs, due to the fact that the Hermite polynomials have zeros at these points. Like a standard GP posterior, the variance is also zero at the observed points (subject to a small nugget term, fixed at 1×10^{-7} here), with uncertainty quantified in between them. When the GP posterior extrapolates outside of the experimental design region, uncertainty is large and the posterior mean function takes the form of the prior mean function multiplied by the appropriate Hermite polynomial.

Figure 5.5 shows the marginal posterior densities for the PC coefficients, a_α, $\alpha = 0, \ldots, 5$, obtained by applying the PPC methodology in Section 5.2.1. The true value of the PC coefficients, obtained by brute force MC integration, is shown as a black line for reference. The estimated value of the PC coefficients using the NISP method for PC with a Gauss-Hermite quadrature rule of size $n = 6$ is also shown as a green line (for a_1, a_3 and a_5 the green and black lines are equal)[2]. Clearly, the posteriors provided by the PPC methodology provide an accurate, probabilistic interpretation of plausible values for the coefficients given the observed simulator runs. In many cases, the maximum a posteriori value of the posteriors are closer to the true value than estimated using the NISP method for PC. The true value also falls within the bulk of the posterior distributions in each case, showing that uncertainty is well quantified. The PPC method also gives a far better estimate of the simulator output mean (given by the zero order coefficient a_0) than traditional PC.

[2]Not intended to be a fair comparison to the PPC method which uses more design points ($n = 10$) — shown for illustrative purposes.

Figure 5.4.: Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a QMC sample from a Sobol sequence of size $n = 10$ (black dots).

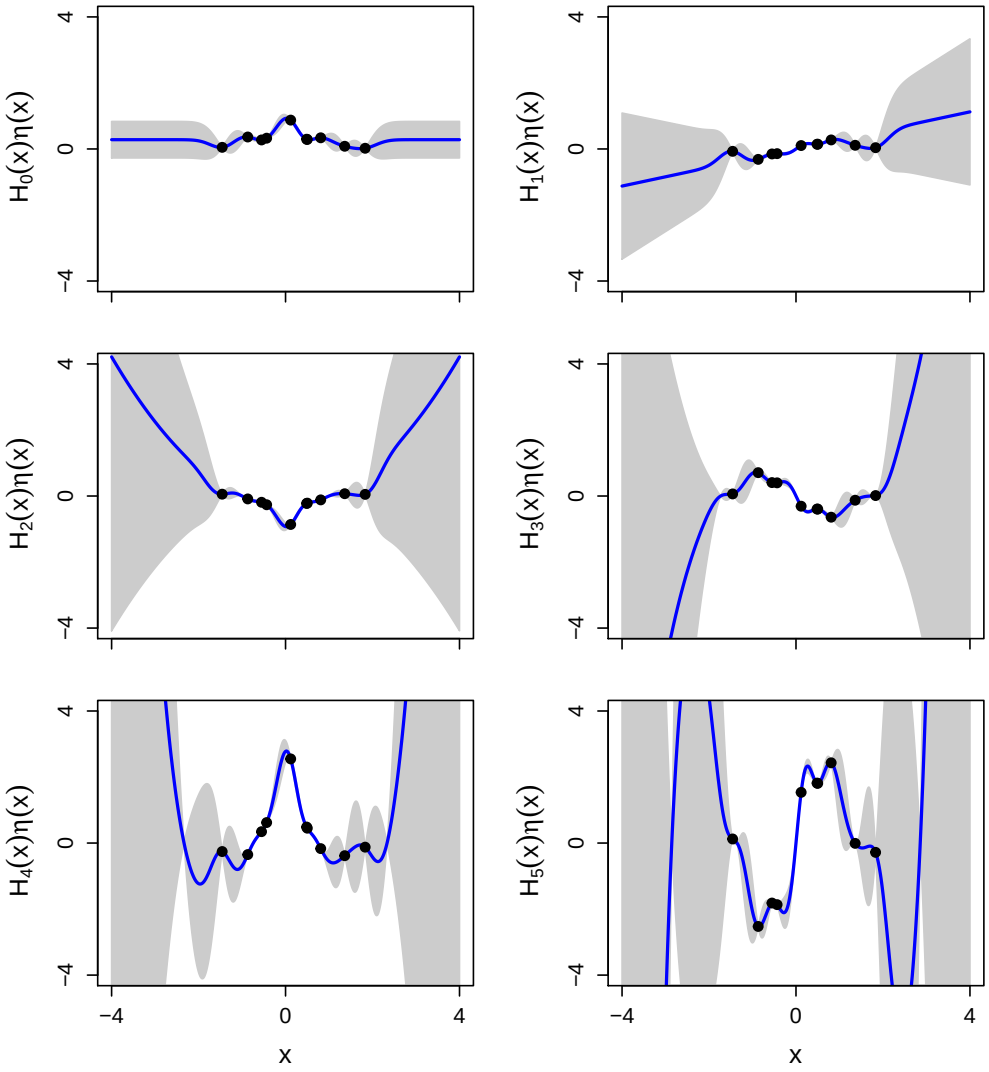


Figure 5.5.: Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 10$ is used. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line.

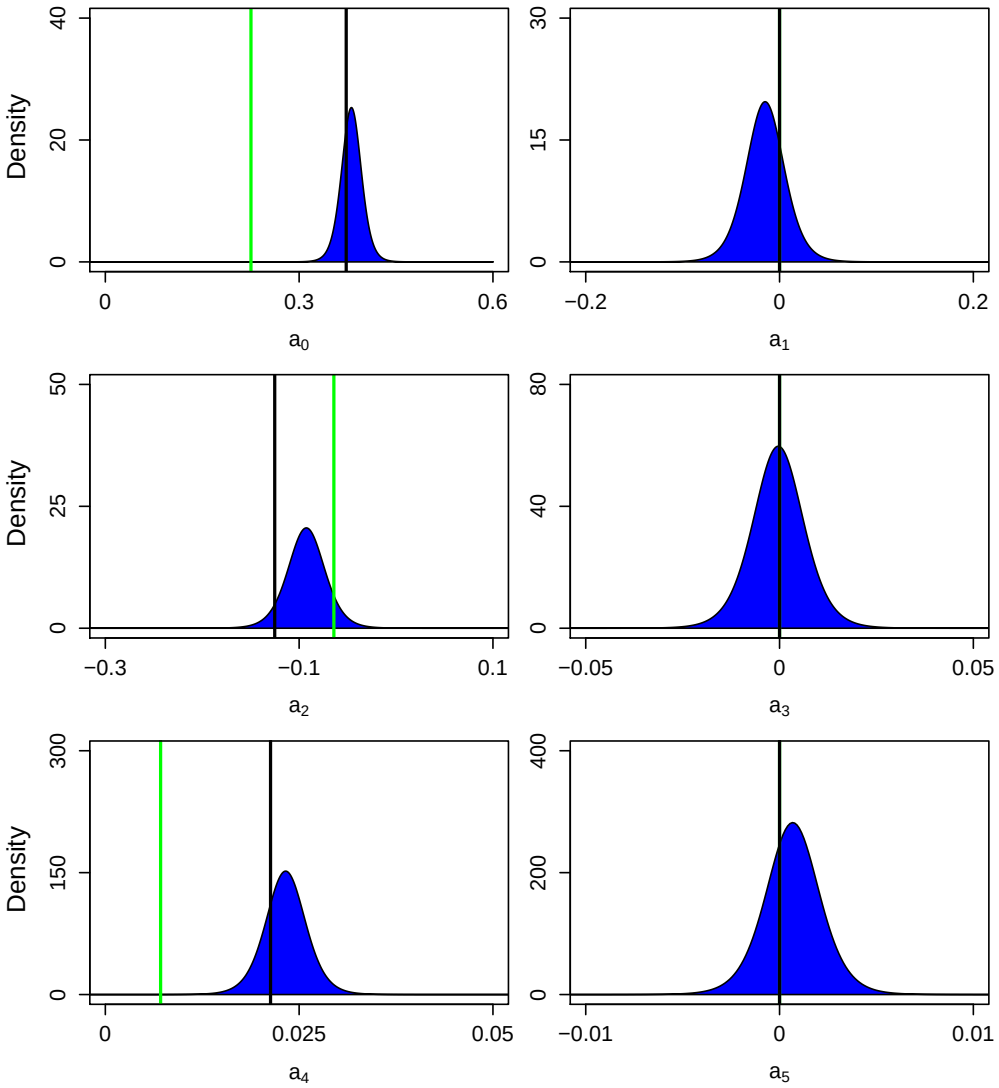


Figure 5.6.: A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 10$ is used.

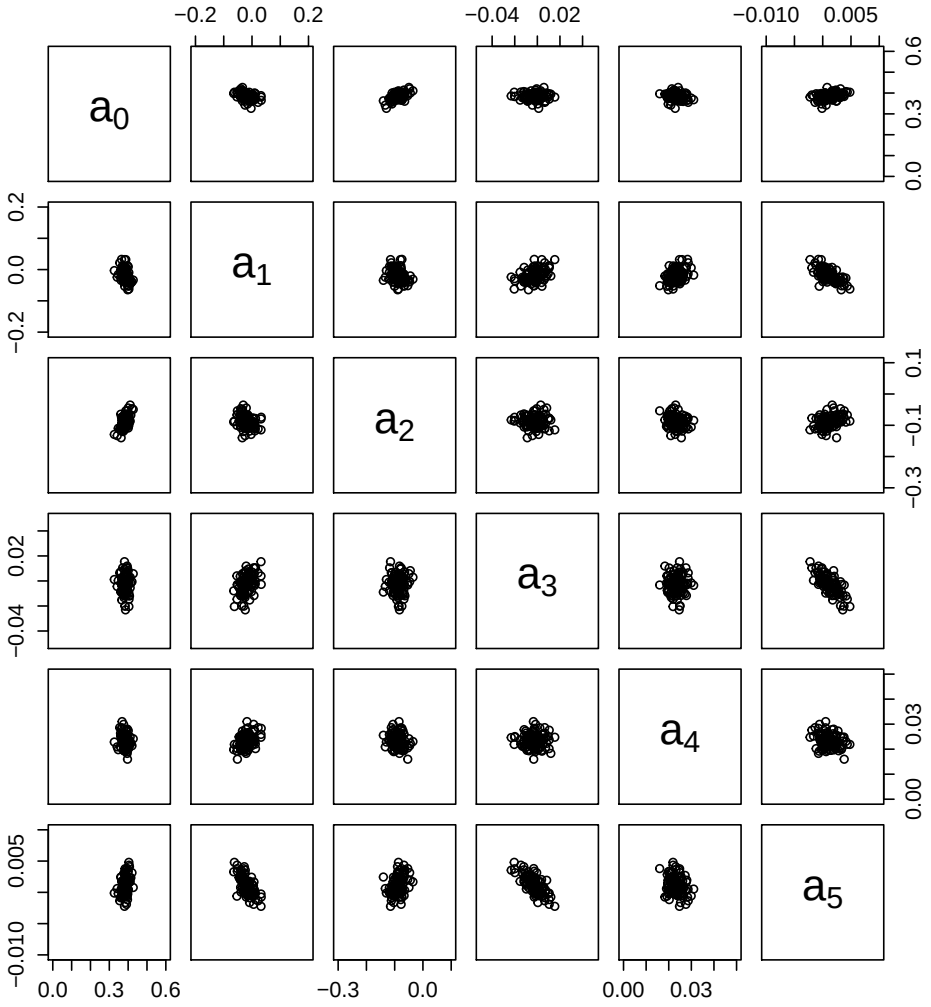


Figure 5.6 shows a sample of size 100 from the joint posterior distribution for the PC coefficients, given by the PPC methodology. This sample shows the extent to which the PC coefficients are correlated with one another. By virtue of the structure of the Hermite polynomials in the truncated PCE, which contain only odd or only even polynomial terms, even (vice versa odd) coefficients are often highly correlated with each other, and different parity coefficients are virtually independent. This would not be the case for a higher dimensional example since the coefficients would be numbered differently. However, the PPC methodology naturally incorporates correlation in the PC coefficients arising from the choice of polynomials.

Finally, Figure 5.7 shows the PPC surrogates, obtained by substituting the posterior for the PC coefficients into the truncated PCE, for each of the design strategies. The top row corresponds to the QMC designs (labelled `QMC`) and the bottom row

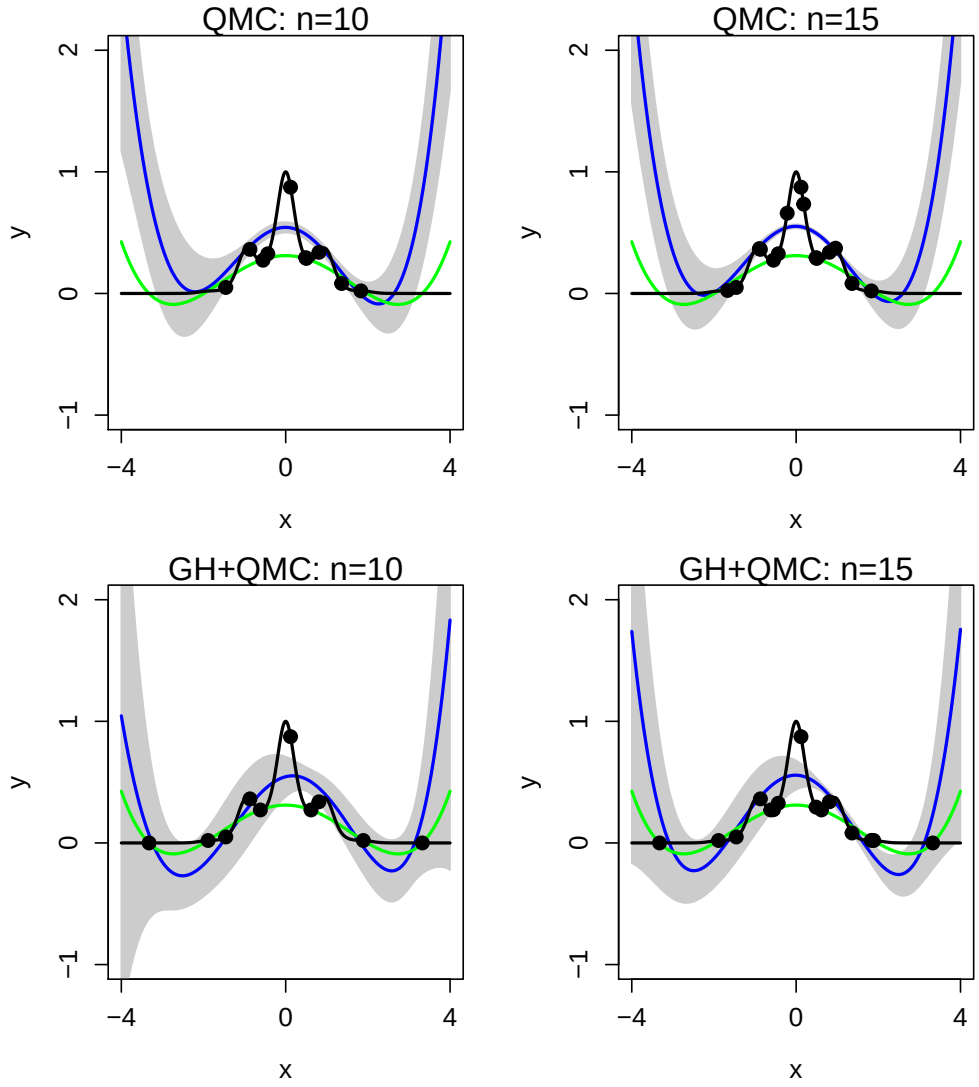
corresponds to the Gauss-Hermite quadrature rules combined with QMC samples (labelled `GH+QMC`). Importantly, the PPC surrogates are not interpolating functions since it is an over-determined problem: the surrogate is restricted to be a fifth order polynomial but design sizes of $n = 10$ and $n = 15$ have been used. This is compared to the traditional NISP for PC approach, where the design size is equal to the number of PC coefficients to be found, giving a uniquely-determined problem and a PC surrogate which interpolates the simulator runs. While the PPC does not necessarily interpolate the simulator runs, it has clearly improved upon the traditional PC surrogate. In all cases featured here, the PPC surrogate is a better predictor for the simulator on average across the input domain. It is also important to remember that the input x is distributed as a standard Gaussian, so the edge effects of the PPC surrogates evident in Figure 5.7 do not have much influence. With regards to which design strategy is best of the two featured, both give similar results in the centre of the input distribution. The only noticeable difference is that the Gauss-Hermite quadrature rule places points in the extreme tails of the input, whereas the QMC sample does not, so the behaviour of the PPC surrogates in the tails of the input distribution is quite different. More research is required to determine the optimal design for PPC.

5.3. Discussion

The PPC methodology presented in this chapter is a promising surrogate approach, combining the emerging field of probabilistic numerics and GP emulators with traditional PC. It is in its early stages of development and the opportunities for further research are vast and wide ranging.

The methodology given in Section 5.2 is applicable to a general simulator $y = \eta(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^d$. However, due to time restrictions, it has only been tested on one-dimensional ($d = 1$) test functions, such as the example in Section 5.2.4. It is hoped in the future that the approach can be tested on higher dimensional problems and simulators used in industry, such as the `adJULES` and `VEGACONTROL` simulators featured in Chapter 4. A potential drawback in scaling the approach to higher dimensions is the curse of dimensionality, where the number of PC coefficients to estimate grows exponentially, and subsequently, the number of design points required becomes very large. However, an important feature of the PPC approach is that a GP emulator for the simulator, $\eta(\mathbf{x})$, only needs to be built once. This emulator is then modified using the appropriate choice of polynomial basis and standard BQ techniques applied. Reliant on the solutions of the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals (which are analytical under the assumptions made here), the posteriors for the PC coefficients can be derived easily and quickly using the simple matrix algebra

Figure 5.7.: Probabilistic polynomial chaos surrogates for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$ for the example in Section 5.2.4. The rows correspond to the two design strategies: Quasi Monte Carlo samples from a Sobol sequence (top row) and Gauss-Hermite quadrature rules combined with Quasi Monte Carlo samples (bottom row). The columns correspond to design sizes of $n = 10, 15$. In each case, the posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The traditional polynomial chaos surrogate, built using a Gauss-Hermite quadrature rule of size $n = 6$ is also shown (green), as well as the true function (black). The experimental design points are shown as black dots.



given in Section 5.2. With regards to the size of the experimental design, if the mean function of the GP is chosen to be a truncated PCE, then the design size must be greater than used for the NISP method for PC, in order for variance information to be obtained. As already mentioned, the design size required for the NISP method for PC is $n = q = (p+1)^d$, which can become prohibitively large in high dimensions. At least a further $d+1$ points are required for estimating the GP hyperparameters, although in high dimensions $d + 1 \ll (p + 1)^d$ so this should not pose a problem. Alternatively, by choosing a simpler mean function (such as a constant mean), the number of GP hyperparameters to estimate is vastly reduced and the burden on design size is eased. There are no restrictions on the type of design for PPC, as design points can be located anywhere in the input domain. Two design strategies were featured in the example in Section 5.2.4, including a promising approach combining traditional Gaussian quadrature rule designs from PC with popular designs from the GP literature, such as the Sobol sequence. It may also be possible to draw upon the sparse grid design literature for PC (Smolyak, 1963; Xiu and Sherwin, 2007), where computationally efficient designs have been proposed for integration in high dimensions. Experimental design for PPC is clearly in its early stages of development and will continue to be a subject of future research.

For the PPC methodology given in Section 5.2, it is assumed that the d inputs to the simulator are identically and independently distributed as standard Gaussian random variables. This results in the use of multivariate Hermite polynomials in the truncated PCE, that is, the classic Wiener-Hermite or Hermite-chaos expansion presented originally by Wiener (1938) and developed by Ghanem and Spanos (1991b). This may not be applicable for a wide range of cases, but the assumption was made for a number of reasons: to simplify and narrow the focus of the work; because the Gaussian distribution was used in the original BQ work of O'Hagan (1991); and mainly because the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals have analytical solutions (needed for the approach to work). A long term goal is to extend the PPC approach for other choices of probability distributions for the inputs of the simulator, and subsequently different orthogonal polynomial families in the truncated PCE. In particular, it is desired to incorporate all probability distributions and polynomial families from the generalised polynomial chaos (gPC) framework (Xiu and Karniadakis, 2003) described in Section 2.3.1 of Chapter 2 and Section 3.2.1 of Chapter 3. Recall the correspondence of orthogonal families from the Askey scheme and input probability distributions for gPC given in Table 2.1 in Chapter 2. The NISP method for traditional gPC uses a tensor product of Gaussian quadrature rules (for which the Gauss-Hermite rule is just one example) appropriate for the probability distributions of the inputs. It is anticipated that the PPC could be extended to include all these cases, since Gaussian quadrature rules can be derived through BQ with the use of the squared exponential correlation function in the GP emulator (O'Hagan,

1991). However, this extension would rely on deriving analytical solutions to the **R**, **T** and **U** integrals for different choices of the probability distribution $f_{\mathbf{X}}(\mathbf{x})$, and the vector of orthogonal polynomials $\boldsymbol{\psi}(\mathbf{x})$ in the truncated PCE (which in the case of gPC would not just be Hermite). Some promising preliminary work has been carried out for the case of uniform probability distributions and Legendre polynomials, but is not shown here due to time restrictions.

The squared exponential correlation function is used in the specification of the GP in the PPC methodology for two reasons: firstly, so that the **R**, **T** and **U** integrals have analytical solutions; and secondly, for a direct interpretation of the Gauss-Hermite quadrature rule. As stated in Chapter 3, the squared exponential correlation function gives a GP which is infinitely differentiable. For some simulators this assumption of smoothness may be too strong and a different choice of correlation function may be more appropriate. In principle any correlation function could be used in the PPC methodology, provided that the **R**, **T** and **U** integrals remain analytical. Briol et al. (2015b) listed combinations of correlation functions and input probability distributions for which standard BQ posteriors can be analytically derived (for example, the Matérn family of correlation functions and the uniform probability distribution). The **R**, **T** and **U** integrals in PPC also include a vector of orthogonal polynomials $\boldsymbol{\psi}(\mathbf{x})$ from the truncated PCE so would require additional work. However, it is expected that results from Briol et al. (2015b) could be combined with repeated integration by parts to give analytical solutions under different choices of the correlation function. This has not been carried out in this work and is an avenue for future research.

Active or sequential design is an important area which has recently been studied in the BQ literature (Osborne et al., 2012a; Gunter et al., 2014; Briol et al., 2015a), where the optimal next integrand evaluation is chosen according to some criteria. While not considered in this work, active learning could be an area of future research for PPC. In this framework, simulator runs could be obtained sequentially or in batches, with the posterior distribution for the PC coefficients updated in stages, until some final accuracy criteria for the PPC surrogate is met. Ideas from the active learning literature for PC could also be applied to PPC, including sparse PC methods which aim to reduce the number of PC coefficients in the expansion, retaining only those with large order of magnitude (Blatman and Sudret, 2011; Jakeman et al., 2015). In turn, this would reduce the computational cost of BQ, since fewer PC coefficients would have to be estimated, and fewer design points would be needed. Another interesting problem is the active selection of the truncation order, p, for the truncated PCE, which is chosen by the user and assumed fixed in this work.

5.4. Conclusion

Despite providing efficient surrogate-based UQ, a key disadvantage of PC as a surrogate technique is the fact that no uncertainty information is provided. While the comparative experiments in Chapter 4 showed that PC and GP emulation performed similarly in terms of approximation accuracy, GP emulators also provide uncertainty information because they are probabilistic surrogates. In particular, a probability distribution for the output is available at input locations where the simulator has not yet been run, quantifying code uncertainty and giving vital information for a practitioner in UQ.

This chapter has provided methodology for overcoming this disadvantage by combining traditional PC with a GP emulator. The method uses techniques from the emerging scientific field of probabilistic numerics, which treats classical numerical methods as statistical inference problems. Focusing on improving the non-intrusive spectral projection method for PC, which estimates the expansion coefficients using the numerical quadrature of intractable integrals, the specific interest of this chapter was on probabilistic approaches to numerical integration. The field of probabilistic numerics, and particularly methods for probabilistic integration, were introduced in Section 5.1. The approach originally introduced by O'Hagan (1991), known as Bayesian quadrature, is especially appropriate for this problem and was discussed in Sections 5.1.2, 5.1.3 and 5.1.4. Using Bayesian quadrature, a posterior distribution for the value of an intractable integral can be derived conditional on a number of integrand evaluations, quantifying the uncertainty induced from a finite computational budget.

The main contribution of this chapter was the application of Bayesian quadrature to the non-intrusive spectral projection method for PC. This novel methodology was presented in Section 5.2 and named probabilistic polynomial chaos. The approach works by applying Bayesian quadrature to the integrals for the coefficients of the PC expansion. The result is a joint posterior distribution for the PC coefficients, quantifying the uncertainty in their estimation from a finite number of simulator runs. This posterior distribution can be substituted into the traditional PC surrogate to give a probabilistic PC surrogate, for use in subsequent UQ analyses.

Due to time restrictions, a number of assumptions were made in developing the probabilistic polynomial chaos approach. In particular, the simulator inputs were assumed to be independent and identically distributed as standard Gaussian random variables, resulting in a probabilistic interpretation of the classic Wiener-Hermite PC expansion, originally developed by Ghanem and Spanos (1990, 1991b). Nevertheless, the application of Bayesian quadrature to this specific case has resulted in a number of interesting mathematical developments, including the novel solution of

the important **R**, **T** and **U** integrals, and the behaviour of a GP emulator when modified with Hermite polynomials. The work presented in this chapter has also raised many questions for the implementation of probabilistic polynomial chaos, as well as several avenues for future research. These were summarised in a discussion in Section 5.3 and include: the expansion of probabilistic polynomial chaos to the generalised polynomial chaos framework of Xiu and Karniadakis (2003); the development of effective experimental designs; the application of existing active learning algorithms from the probabilistic numerics, GP emulation and PC literatures; and finally, the application of probabilistic polynomial chaos to higher dimensional problems and simulators used in industry.

6. Conclusion

This chapter gives a summary of the key findings of this thesis and suggests a number of directions for future development of the work.

6.1. Summary

In Chapter 2, a literature review of the developments of polynomial chaos and Gaussian process emulation over the last 25 years was conducted, and a number of points were raised. In particular, the two approaches have been developed largely independently in the engineering and statistics communities respectively. Despite both methods providing efficient solutions for surrogate-based uncertainty quantification of expensive simulators, there has been a surprising little amount of interaction and collaboration between the two communities. An effort to bring the two communities closer together was made by O'Hagan (2013), who gave a tutorial and critique of polynomial chaos from a statistician's perspective. However, this work was never published. Only recently have there been examples comparing and combining polynomial chaos and Gaussian process emulation, but these are still rare in the literature. An indirect comparison of the two methods was given by Liu et al. (2017), who tested gradient-enhanced surrogates against quasi-Monte Carlo simulation. A hybrid approach was presented by DiazDelaO and Adhikari (2009, 2011), who coupled a Gaussian process emulator with a traditional intrusive framework for polynomial chaos (Ghanem and Spanos, 1991b) to reduce the computational burden. A promising non-intrusive hybrid approach called PC-Kriging was given by Schöbi et al. (2015), which added a zero-mean Gaussian process to a polynomial chaos surrogate built adaptively using regression, and this has been applied successfully in a number of examples (Kersaudy et al., 2015; Schöbi and Sudret, 2015; Schöbi et al., 2016). A key conclusion from Chapter 2 was the need for a direct critical comparison of the two methods for a range of criteria, as it is unclear to the uncertainty quantification community as a whole which method may perform better in different modelling scenarios. Furthermore, there is clear scope for development of hybrid surrogate techniques different to those already proposed.

In Chapter 3, the notation for surrogate modelling was introduced, followed by a

detailed mathematical description of polynomial chaos and Gaussian process emulation. Special care was taken to avoid presenting advanced techniques for either of the methods, focussing rather on versions used commonly for applications in the uncertainty quantification literature. This was intended to begin a comparison of the two methods from the ground up, and provide a foundation for future studies. For polynomial chaos, the truncation order, p, of the polynomial chaos expansion was chosen a priori, and two schemes for truncating the expansion were featured: total order and tensor product truncation (Eldred and Burkardt, 2009). Furthermore, two non-intrusive methods for estimating the coefficients of the polynomial chaos expansion were given: regression (Blatman and Sudret, 2011) and non-intrusive spectral projection using numerical quadrature (Reagan et al., 2003). In terms of Gaussian process emulation, the prior mean function was restricted to be a linear model and two stationary, separable covariance functions were considered: the squared exponential, and the Matérn with shape parameter set to $\nu = 5/2$ (Rasmussen and Williams, 2006). When deriving the posterior Gaussian process, a 'plug-in' treatment of the correlation length hyperparameters was considered, as advocated by Kennedy and O'Hagan (2001) and subsequently used in much of the Gaussian process literature (Oakley and O'Hagan, 2002, 2004; Bayarri et al., 2007). With the methodology introduced, polynomial chaos and Gaussian process surrogates were applied to simple one-dimensional and two-dimensional examples for demonstrative purposes. For these examples it was evident that the two surrogate approaches gave quite different approximations to the test functions, even when built on the same designs. This motivated the need for a more in depth, critical comparison of the two methods in terms of their approximation accuracy, as well as other criteria, such as their flexibility and practicality.

Chapter 4 was concerned with the validation of surrogate models. It was shown that techniques for assessing the quality of a surrogate take different forms in the polynomial chaos and Gaussian process emulation communities respectively, which perhaps has prohibited a comparison of the two methods in the literature. To allow such a comparison, a set of validation metrics were proposed which relied on an additional set of simulator runs in a validation design. These included the root mean square error, which gave a single measure of the accuracy of the surrogate, as well as metrics testing the ability of the surrogate in estimating summary statistics of the simulator output: the mean, standard deviation, exceedance probabilities and the probability density function. Particular interest was in examining how the comparative accuracy of the surrogates, measured using the validation metrics, changed with variations in the size and type of the experimental design. Experiments were conducted for two simulators used in industry: `adJULES`, an environmental simulator describing the interactions between the land and the atmosphere, and `VEGACONTROL`, a simulator modelling the flight dynamics of the VEGA launch vehicle.

At the time of writing, this is the first direct comparison of polynomial chaos and Gaussian process emulation in the literature, with results published in Owen et al. (2017).

In short, it was found that the preferred surrogate method depended on the modelling goals of the practitioner. More specifically, in terms of the root mean square error, polynomial chaos surrogates were preferred for tensor grid designs, whereas Gaussian process emulators were favoured on less structured designs such as Sobol sequences. However, it was also shown that the performance gap between polynomial chaos and Gaussian process surrogates built on large tensor grid designs could be completely eliminated, when instead building the surrogates on a Sobol sequence design of the same size. When estimating statistical summaries of the simulator output, both surrogates proved highly accurate for estimating the mean and standard deviation, even for small design sizes. Some preference for polynomial chaos could be found here for the `adJULES` experiment, and this was also the case when estimating exceedance probabilities. When estimating the probability density function of the simulator output, there was little difference between the surrogate approaches. For all validation metrics, a clear preference for Gaussian process emulation could be found for class 1 designs, suggesting that they are more suitable when faced with very small design sizes.

The results from the experiments in Chapter 4, as well as the methodology from Chapter 3, also allowed some general statements to be made about the comparative advantages and disadvantages of each approach. In terms of the computational cost of building the surrogate, polynomial chaos has the edge in that it just relies on plug-in formulas (quadrature or matrix algebra) to estimate the coefficients of the expansion, after the selection of the orthogonal polynomial basis. On the other hand, the computational cost of building a Gaussian process emulator stems from the estimation of the correlation length hyperparameters, which involves the repeated inversion of the correlation matrix. Regarding the flexibility of the surrogate models, Gaussian process emulators were shown to be better for two reasons. Firstly, there are no restrictions on the size and type of experimental design to build a Gaussian process emulator; whereas for polynomial chaos a certain design size is required to ensure a specific truncation order, and for the non-intrusive spectral projection method the design itself must be a Gaussian quadrature rule. Secondly, Gaussian process emulators can model a much wider range of simulator behaviours — not just polynomial functions — and this can be done relatively easily with changes in the mean and covariance functions. This was demonstrated by repeating the experiments for a non-linear simulator with two inputs. Concerning the practicality of the surrogates, a key issue is the fact that the polynomial chaos surrogate does not provide uncertainty information with its predictions. On the contrary, Gaussian

process emulators are fully probabilistic surrogates, quantifying code uncertainty where the simulator has not yet been run.

Chapter 5 presented a novel hybrid methodology called probabilistic polynomial chaos, which combined polynomial chaos with Gaussian process emulators. The approach drew inspiration from an emerging field in scientific computation known as probabilistic numerics, which treats classical numerical methods as statistical inference problems. Specifically, a probabilistic integration method called Bayesian quadrature (O'Hagan, 1991), was adapted and applied to the non-intrusive spectral projection method for polynomial chaos. By placing a Gaussian process prior on the relevant integrands, it was shown how a posterior distribution for the coefficients of the polynomial chaos expansion could be derived, conditional on the simulator runs. This posterior distribution quantifies the uncertainty in the estimation of the coefficients from a finite set of simulator evaluations. Substituting the uncertain coefficients into the polynomial chaos expansion resulted in a probabilistic interpretation of the traditional polynomial chaos surrogate. This rectifies the key disadvantage of polynomial chaos, namely the lack of uncertainty information provided, outlined in Chapter 4. Novel contributions were also uncovered in the implementation of probabilistic polynomial chaos, including: the solution of the important $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals; the behaviour of a Gaussian process emulator modified by Hermite polynomials; how the traditional non-intrusive spectral projection method could be recovered as a special case of probabilistic polynomial chaos; and possible experimental designs for effective estimation of the variance hyperparameters of the Gaussian process emulator.

6.2. Directions for future development

There are several possible directions for future development of the work presented in this thesis. Firstly, the comparisons of polynomial chaos and Gaussian process emulation in Chapter 4 could be extended in various ways. The experiments were primarily designed to test the comparative approximation accuracy of the surrogates under changes in the size and type of the experimental design. Three design classes were used, in which different non-intrusive techniques for estimating the coefficients of the polynomial chaos expansion were implemented. Each design class featured four distinct designs of increasing size, to ensure polynomial chaos expansions with truncation orders $p = 1, 2, 3, 4$. Due to computational and time restrictions, it was only possible to obtain one realisation of these designs, as well as a single validation design for computing the validation metrics. This is particularly relevant for design classes 1 and 2, which were random samples from a Sobol sequence design. One possible future study would be to examine the effect of sampling variability on the

approximation accuracy of the surrogates, that is, repeat the experiments using other random samples of the same size.

Two simulators used in industry were implemented, `adJULES` and `VEGACONTROL`, with a single output and relevant set of inputs chosen through expert elicitation in each case. A nice feature was that the conclusions drawn from the experiments were seen to hold for both simulators. However, it would no doubt be beneficial to compare the surrogates for different formulations of the simulators, or for other simulators entirely. In particular, since the input dimension for both simulators was chosen to be relatively low — $d = 4$ and 5 in the `adJULES` and `VEGACONTROL` cases respectively — it would be interesting to see how the surrogates compare for higher dimensional problems. Moreover, there is scope to compare the surrogates in scenarios where the simulator output is also high dimensional (for example, a time series or spatial field), since the assumption of a single, scalar, simulator output was made throughout this thesis for simplicity. In both of these cases, the methodology of both polynomial chaos and Gaussian process emulation would have to be adapted accordingly.

Pertinent to this, there is also scope for a comparison of the methods in their more advanced formulations. As already mentioned, in Chapter 3 effort was made to present versions of polynomial chaos and Gaussian process emulation that are commonly used in the uncertainty quantification literature. This was to facilitate a comparison of the methods from the ground up, but also to provide useful information for practitioners as to which might be better in certain scenarios. A (not exhaustive) list of more advanced, or alternative, approaches for each method includes: sparse polynomial chaos (Blatman and Sudret, 2011; Jakeman et al., 2015), non-intrusive spectral projection on sparse grids (Xiu and Sherwin, 2007), local polynomial chaos (Le Maître et al., 2004a; Wan and Karniadakis, 2005), arbitrary polynomial chaos (Soize and Ghanem, 2004), and stochastic collocation (Xiu and Hesthaven, 2005); Bayes Linear (Craig et al., 2001; Goldstein and Rougier, 2006) or fully Bayesian (Neal, 1998; Higdon et al., 2008) emulators, nonstationary emulators (Sampson and Guttorp, 1992; Gramacy and Lee, 2008; Ba and Joseph, 2012), and emulators for large computer experiments (Gramacy and Apley, 2015; Gu and Berger, 2016). A critical comparison of the surrogates in these formulations may reveal which is better in more specific modelling scenarios.

Secondly, the method of probabilistic polynomial chaos, proposed in Chapter 5, is still in its early stages of development and there is much scope for future work. The assumption that the inputs are independently and identically distributed as standard Gaussian random variables meant that the proposed method is a probabilistic version of the classic Wiener-Hermite expansion, originally given by Ghanem and Spanos (1991b). A long term goal is to extend the probabilistic polynomial chaos methodology to include all possible types of probability distributions for the in-

puts, and hence orthogonal polynomial families, in the generalised polynomial chaos framework (Xiu and Karniadakis, 2003). Whether this is possible will largely boil down to whether analytical solutions of the $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{U}$ integrals are available for all appropriate choices of probability distribution and orthogonal polynomial basis. It was conjectured that this will indeed be possible, through the application of integration by parts in conjunction with theoretical results from Briol et al. (2015b). Promising preliminary work has been carried out for the case of uniform distributions and Legendre polynomials, but was not featured in this thesis due to time constraints.

The probabilistic polynomial chaos surrogate was only tested on a simple one-dimensional example. It is therefore of interest how the novel methodology scales to higher dimensional test functions and simulators used in industry. It is expected that the method will scale well, even though the number of polynomial chaos coefficients to estimate becomes large in high dimensions, since only one Gaussian process emulator needs to be built before being modified by the appropriate polynomial basis and applying Bayesian quadrature. Also related to the scalability of the approach is the size of the experimental design required to built the surrogate. When the mean function of the Gaussian process is chosen to be a truncated polynomial chaos expansion itself, the number of design points needed is large in high dimensions. Furthermore, extra design points are needed to estimate the variance hyperparameters of the Gaussian process. In this case, a possible design strategy is the combination of traditional quadrature rules used in polynomial chaos with designs suited to estimating the hyperparameters of the Gaussian process emulator, for example Sobol sequence or Latin Hypercube designs. However, if this number of design points cannot be obtained from the expensive simulator, an alternative is to simplify the form of the mean function to reduce the computational burden. Since any type of design may be used in theory, efficient experimental design for probabilistic polynomial chaos surrogates remains an open question. A potential strategy could be to combine probabilistic polynomial chaos with techniques in polynomial chaos for high dimensional quadrature, such as quadrature on sparse grids (Smolyak, 1963; Xiu and Sherwin, 2007). Alternatively, ideas from the active learning literature could be employed, such as algorithms for active or sequential design in Bayesian quadrature (Osborne et al., 2012a; Gunter et al., 2014; Briol et al., 2015a) or sparse polynomial chaos (Blatman and Sudret, 2011).

6.3. Conclusion

This thesis has discussed surrogate-based approaches for uncertainty quantification in computer experiments, with particular focus on polynomial chaos and Gaussian

process emulation. Firstly, a critical comparison of the approximation accuracy of two methods was given for two simulators used in industry, highlighting their respective advantages and disadvantages, and providing useful advice for practitioners in uncertainty quantification. Secondly, a novel hybrid methodology combining polynomial chaos and Gaussian process emulation was proposed, called probabilistic polynomial chaos. It is hoped that this work provides the foundation for further comparisons of the two methods, as well as further development of the probabilistic polynomial chaos methodology, and ultimately, sparks more interaction, discussion and collaboration between the polynomial chaos and Gaussian process emulation communities.

Appendices

A. Additional figures for the probabilistic polynomial chaos examples

Figure A.1.: Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a QMC sample from a Sobol sequence of size $n = 15$ (black dots).

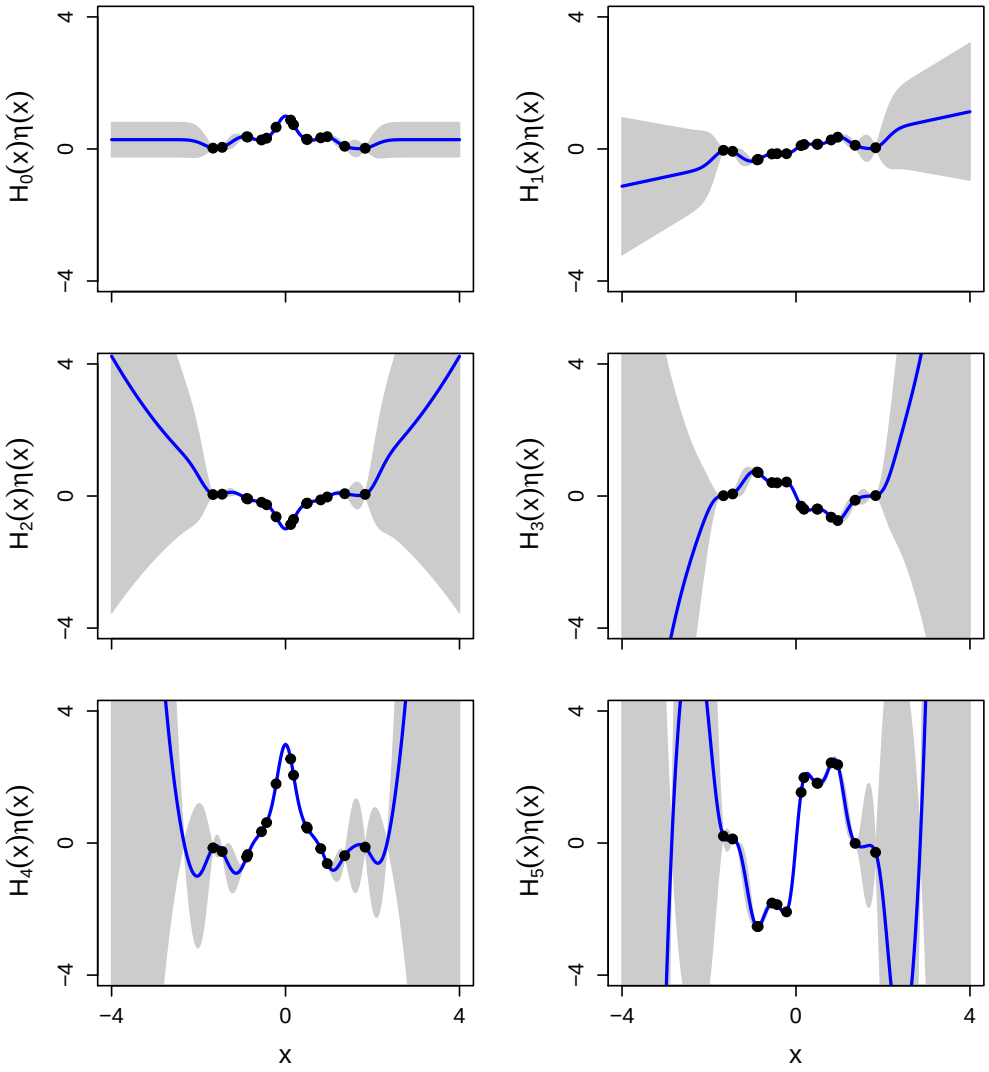


Figure A.2.: Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 15$ is used. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line.

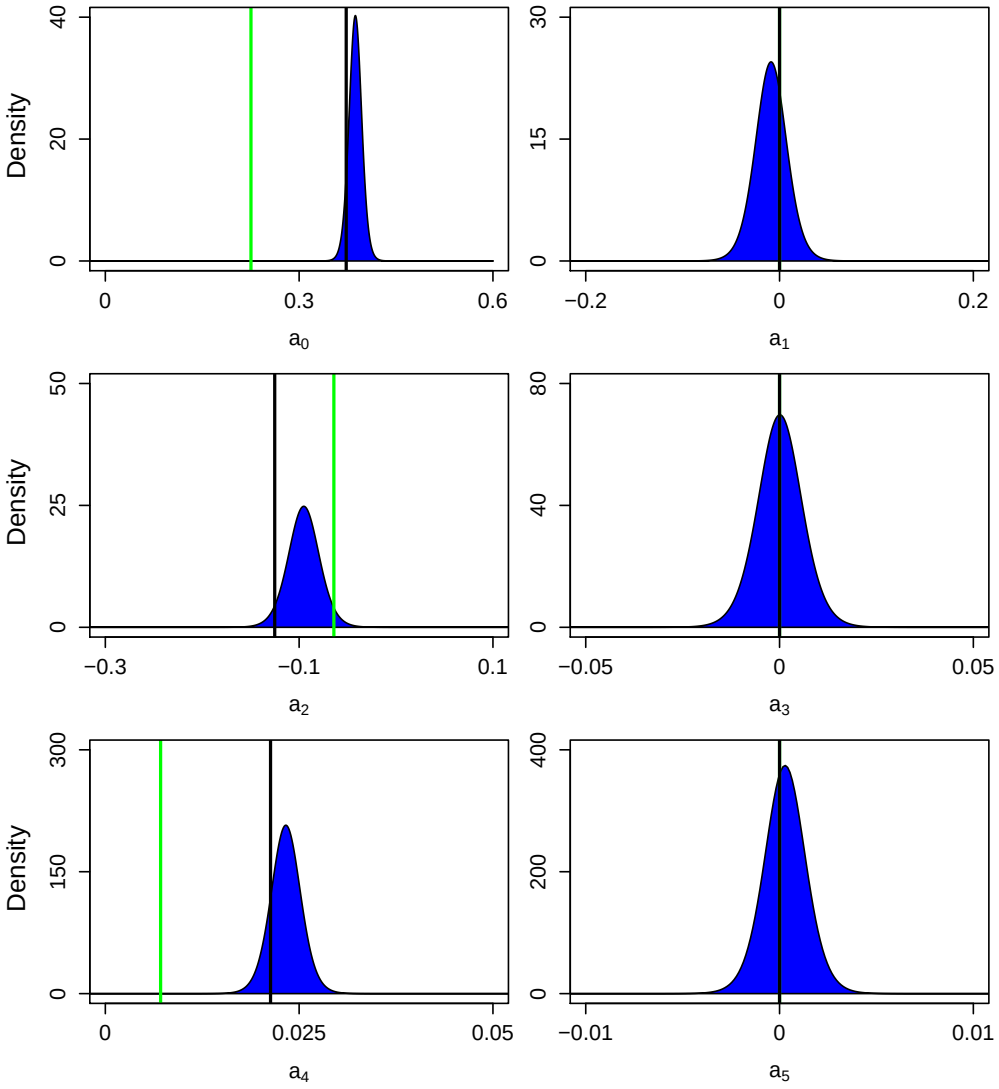


Figure A.3.: A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A QMC sample from a Sobol sequence of size $n = 15$ is used.

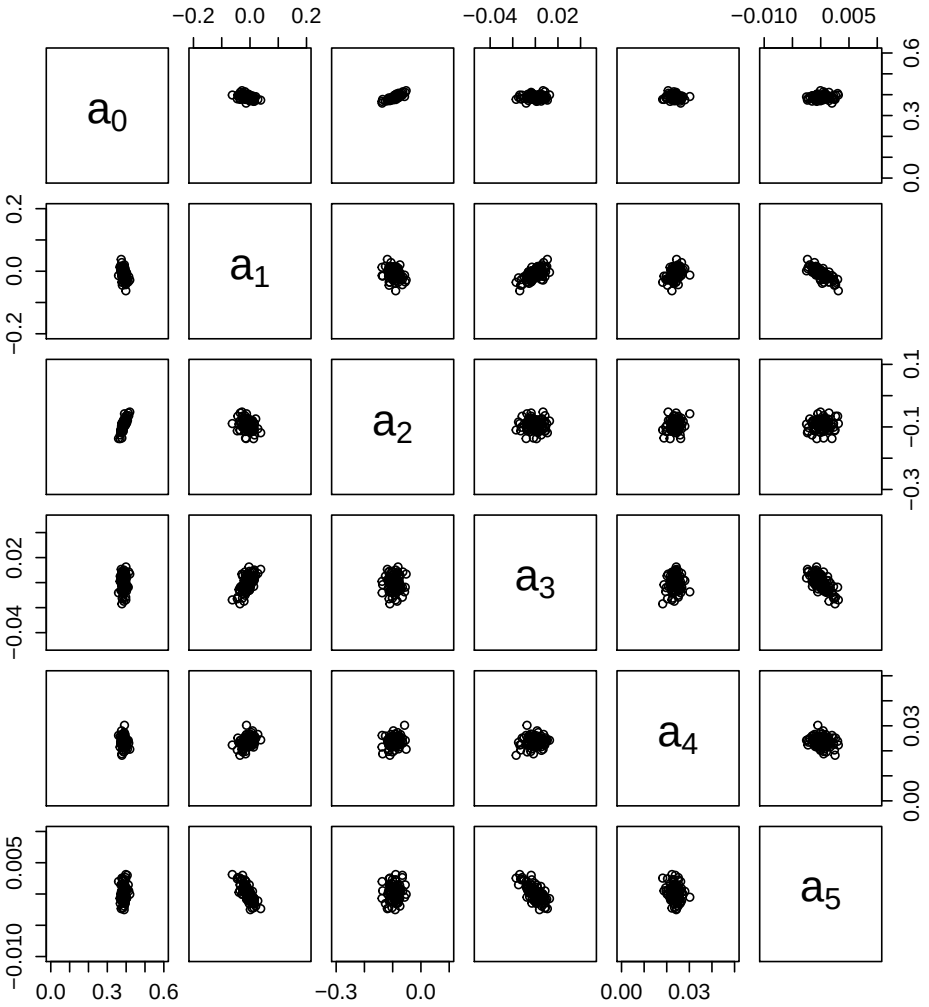


Figure A.4.: Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4, to give a design size $n = 10$ (black dots).

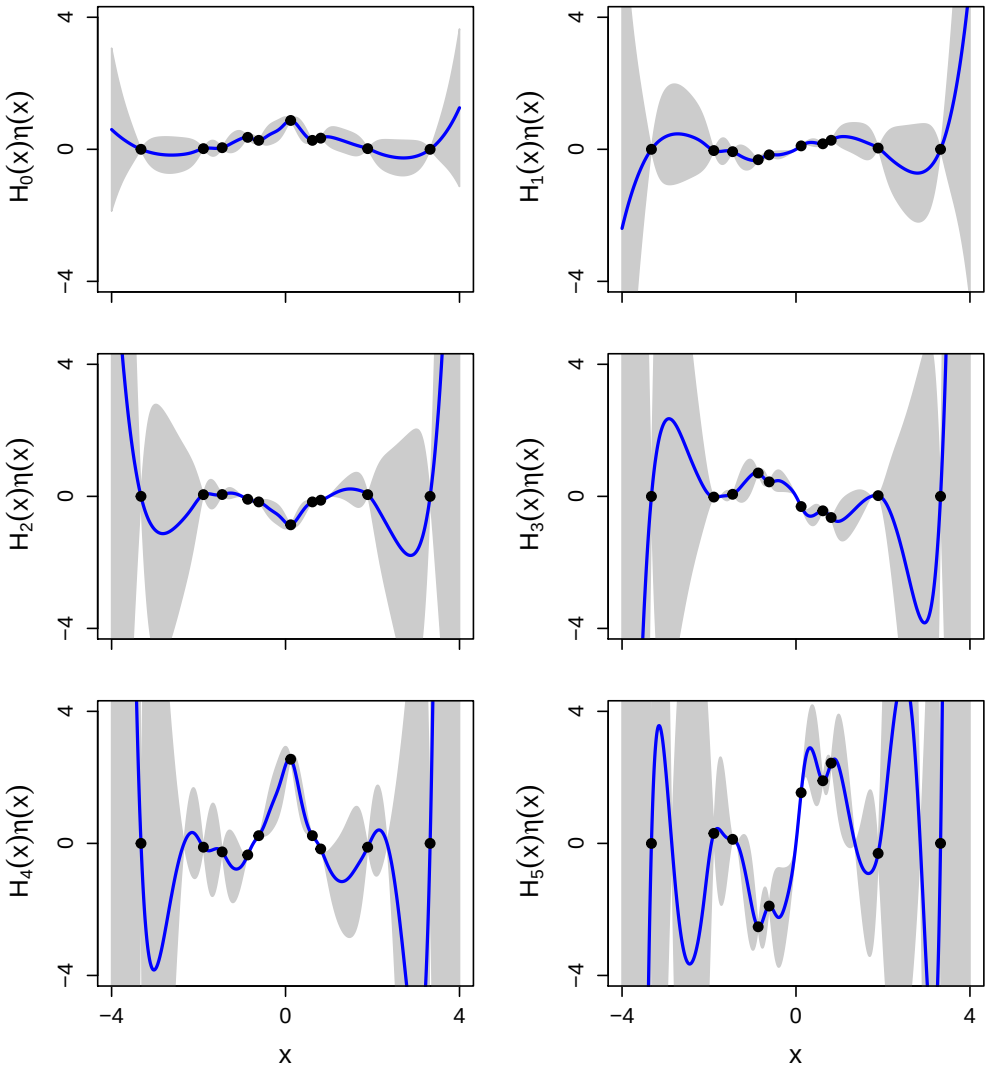


Figure A.5.: Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4 is used, to give a design size $n = 10$. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line.

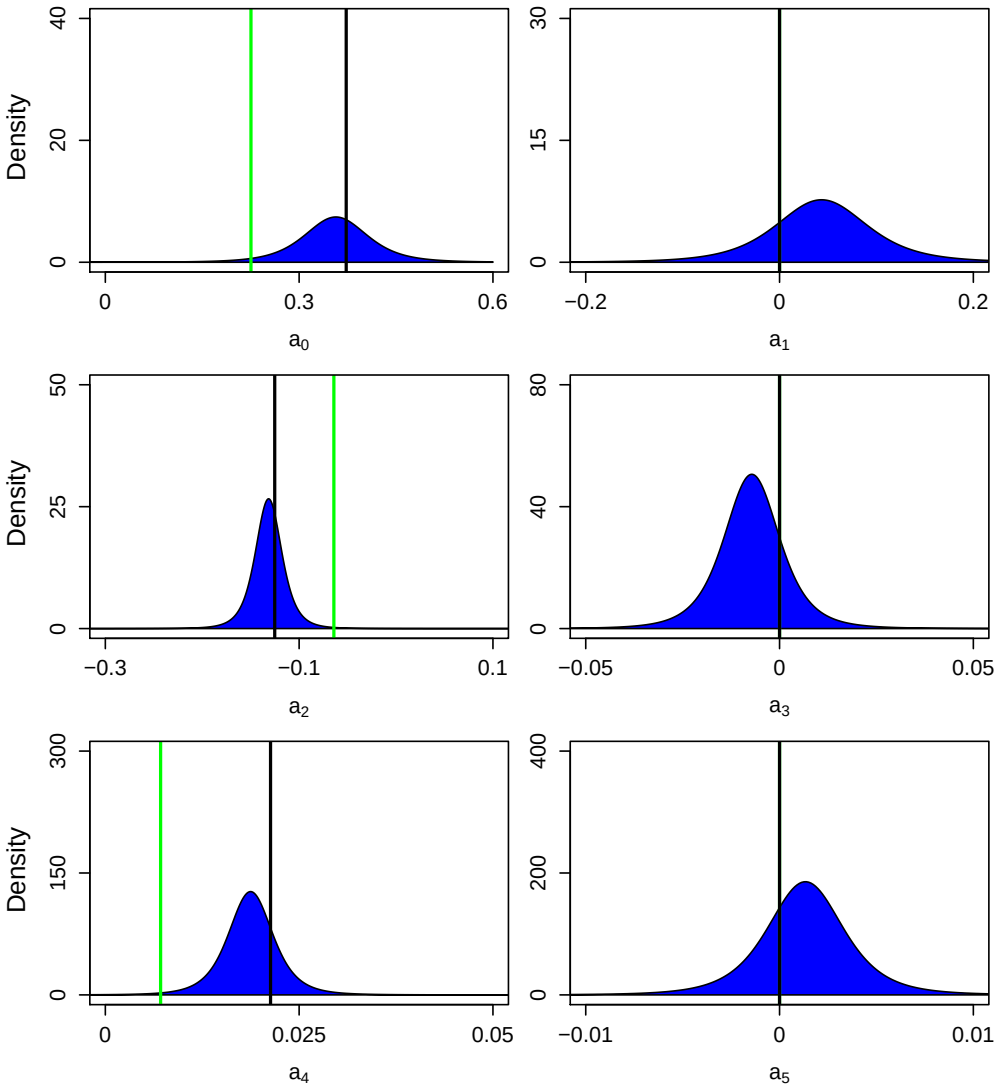


Figure A.6.: A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 4 is used, to give a design size $n = 10$.

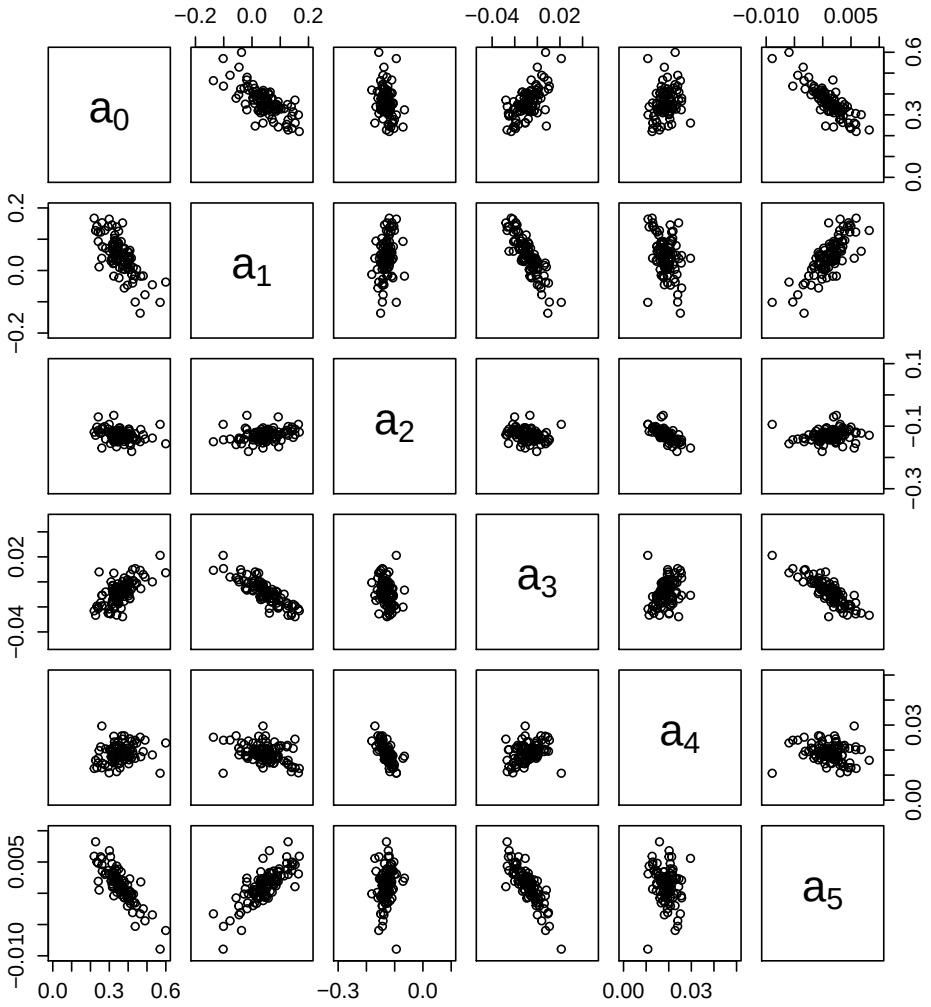


Figure A.7.: Gaussian process posteriors for the function $y = \eta(x) = \exp(-\sin^2(3x) - x^2)$, modified by the Hermite polynomials $H_\alpha(x)$, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. The posterior mean function is shown (blue) with a posterior uncertainty represented as a region of two standard deviations around the mean (grey shading). The experimental design points are a Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9, to give a design size $n = 15$ (black dots).

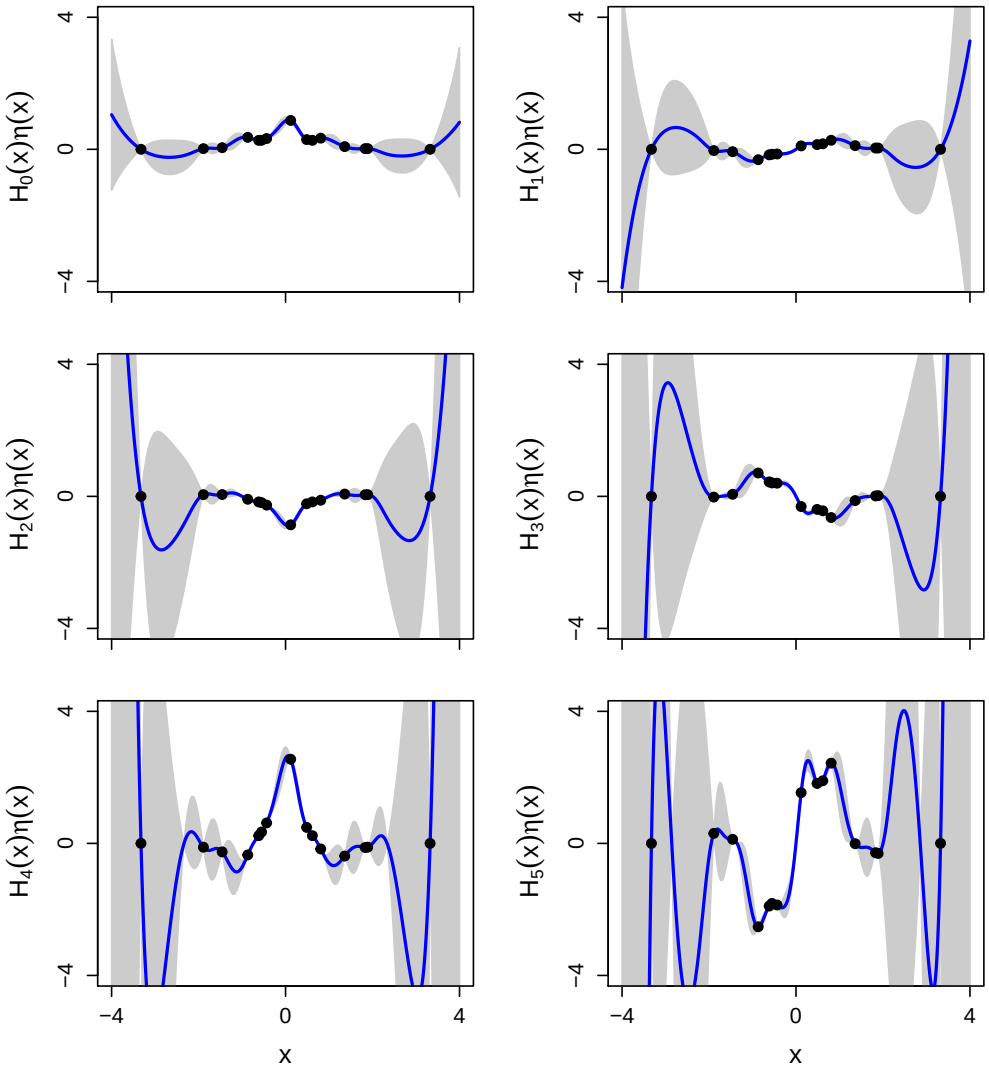


Figure A.8.: Marginal posterior densities (blue) given probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9 is used, to give a design size $n = 15$. The true value of the coefficients, given by brute force Monte Carlo integration, is shown as a black line. The estimated value of the coefficients from traditional polynomial chaos is shown as a green line.

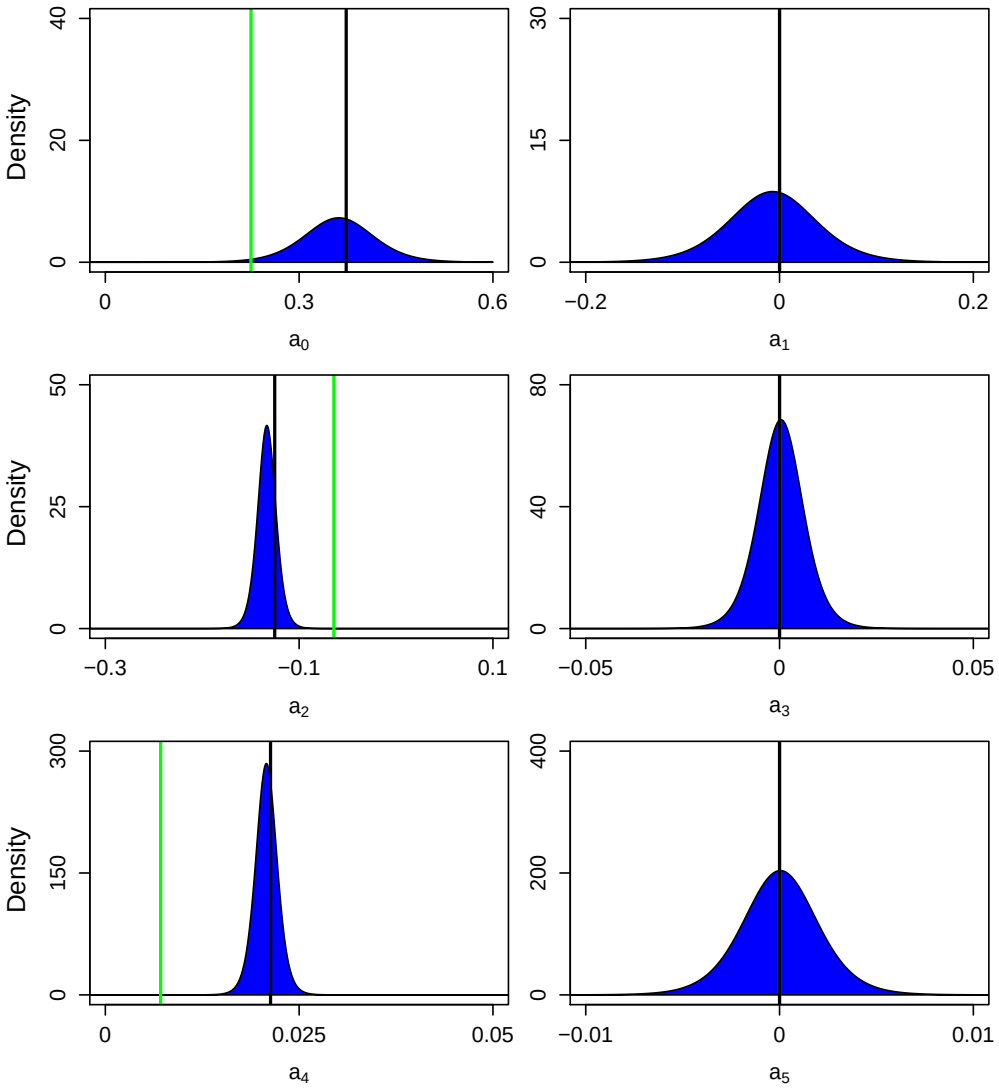
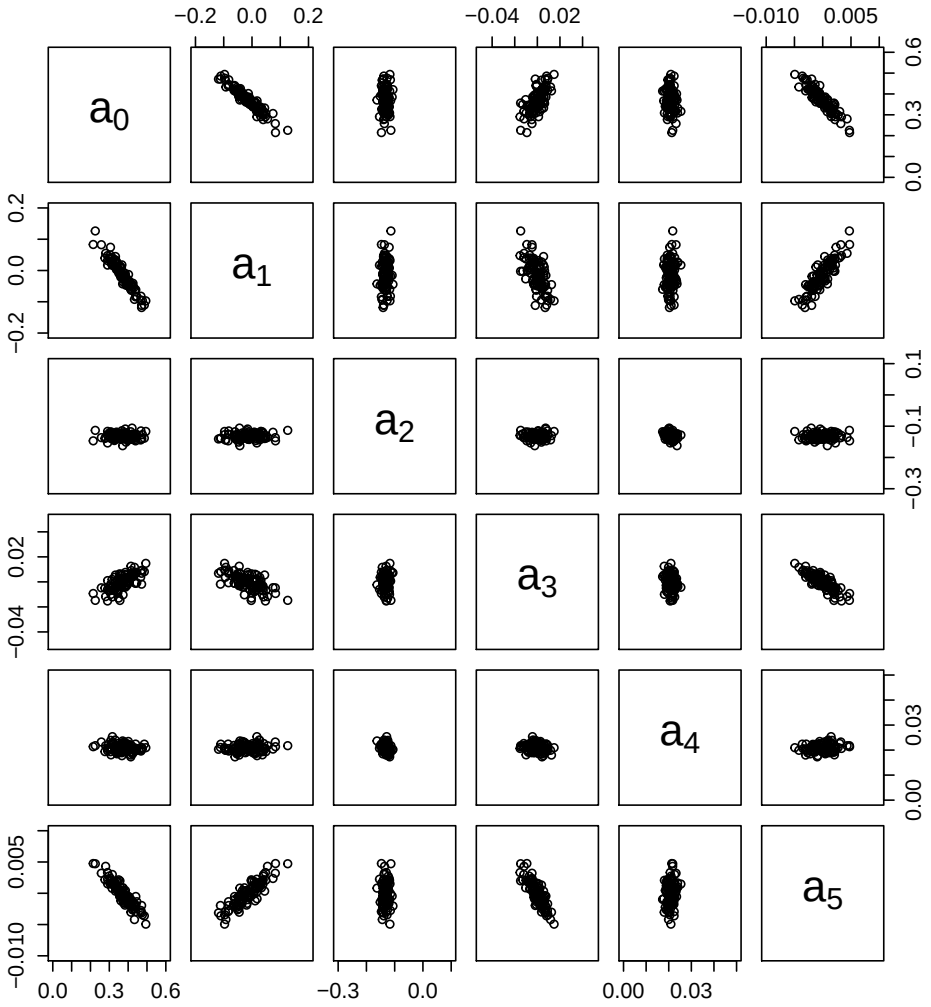


Figure A.9.: A sample of size 100 from the joint posterior distribution given by probabilistic polynomial chaos for the polynomial chaos coefficients a_α, $\alpha = 0, \ldots, 5$, for the example in Section 5.2.4. A Gauss-Hermite quadrature rule of size 6, combined with a QMC sample from a Sobol sequence of size 9 is used, to give a design size $n = 15$.



B. Comments from the viva voce examination with response from the author

In this appendix, comments and suggestions raised by the examiners in the viva voce examination of this thesis will be presented and discussed. Note that the mathematical notation deviates from that used in the main bulk of the thesis.

B.1. Chapter 1

No comments given.

B.2. Chapter 2

Theoretical connection between Gaussian process emulation and polynomial chaos

The examiners suggested that a theoretical connection between Gaussian process (GP) emulation and polynomial chaos (PC) could perhaps be explored in more detail. The following approach was presented and discussed.

Start with a GP with mean 0 (without loss of generality) and covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Denote the reference measure on $\mathcal{X}$ as $f_{\mathbf{X}}$. The aim below is to start with a GP and see if something similar to PC can be recovered. Mercer's theorem gives the following representation (under mild conditions):

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=0}^{\infty} \lambda_j e_j(\mathbf{x}) e_j(\mathbf{x}'),$$

where λ_j and e_j are eigenvalues and eigenfunctions of the covariance operator $\Sigma_k : L^2(f_{\mathbf{X}}) \to L^2(f_{\mathbf{X}})$ associated to the covariance function k. The covariance function

k is associated with a native space of functions, a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ whose elements are:

$$\mathcal{H} = \left\{ \eta(\mathbf{x}) = \sum_{j=0}^{\infty} c_j \lambda_j^{1/2} e_j(\mathbf{x}) : \|\eta\|_{\mathcal{H}}^2 = \sum_{j=0}^{\infty} c_j^2 < \infty \right\}.$$

A sample η from the GP can be obtained by letting $c_j \sim N(0,1)$ be independent[1] (Rasmussen and Williams, 2006).

A reduced rank approximation to a GP is an approximation of the form:

$$\eta_m(\mathbf{x}) = \sum_{j=0}^{m} c_j \lambda_j^{1/2} e_j(\mathbf{x}),$$

where again $c_j \sim N(0,1)$ are independent. Reduced-rank approximations are widely used in classical statistics due to their simple interpretation as Bayesian linear regression models. Indeed, the approximation error can be bounded as:

$$\|\eta - \eta_m\|_{\mathcal{H}} = \left\| \sum_{j=0}^{m} c_j \lambda_j^{1/2} e_j(\mathbf{x}) \right\|_{\mathcal{H}} = \left(\sum_{j=m+1}^{\infty} c_j^2 \right)^{1/2},$$

so analysis is straightforward. Moreover, an estimator for the coefficients c_j can be obtained with least squares, at a computational cost that is gated by m rather than the number $n \gg m$ of data $\{(\mathbf{x}_i, \eta(\mathbf{x}_i))\}_{i=1}^n$.

Now, consider $\mathcal{X} = \mathbb{R}$, $f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$ and $k(x,x') = \exp(-\frac{(x-x')^2}{l^2})$. For these choices, Shi et al. (2009) showed that:

$$\lambda_j = \sqrt{\frac{2}{1+\beta+\sqrt{1+2\beta}}} \left(\frac{\beta}{1+\beta+\sqrt{1+2\beta}} \right)^j,$$
$$e_j(x) = \frac{(1+2\beta)^{1/8}}{\sqrt{2^j j!}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \frac{\sqrt{1+2\beta}-1}{2} \right) H_j\left(\left(\frac{1}{4} + \frac{\beta}{2} \right)^{1/4} \frac{x-\mu}{\sigma} \right),$$

where H_j are the Hermite polynomials. Then the reduced-rank approximation to the GP has the form:

$$\eta_m(\mathbf{x}) = \sum_{j=0}^{m} a_j \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \frac{\sqrt{1+2\beta}-1}{2} \right) H_j\left(\left(\frac{1}{4} + \frac{\beta}{2} \right)^{1/4} \frac{x-\mu}{\sigma} \right),$$

where the coefficients $a_j \sim N(0, \gamma_j)$ are independent for some γ_j, whose definition here is implicit. For $\beta \ll 1$ we can approximate the exponential term as 1 and thus

[1] Note that a sample from the GP lies outside $\mathcal{H}$ with probability 1.

we obtain:

$$\eta_m(\mathbf{x}) \approx \sum_{j=0}^{m} a_j H_j \left(\frac{x-\mu}{\sigma\sqrt{2}} \right).$$

This closely resembled the PC approximation method. This toy theoretical framework gives some hope that deeper connections between GP and PC can be established.

An alternative approach to connect PC to GP is to construct the covariance function k based on Hermite polynomials to generate a finite-dimensional RKHS, the details of which are given by Särkkä et al. (2016).

Response

This thesis was mainly concerned with the comparison of PC and GP methods in their "off-the-shelf" forms, and focused on a direct comparison in terms of their approximation accuracy for simulators used in industry, to be of most use to practitioners in uncertainty quantification. However, the toy example presented above shows that it may possible to prove a direct theoretical connection between the two methods. Using Mercer's theorem to decompose a GP covariance function in terms of eigenvalues and eigenvectors, it seems that the resulting reduced-rank representation of a GP closely resembles a PC expansion. In this way, it may be possible to derive theoretical results on the approximation accuracy of either method and compare them to the empirical results obtained in Chapter 4. It may also provide further insight for the proposed probabilistic polynomial chaos method in Chapter 5.

The toy example presented considers a squared exponential covariance function and a Gaussian probability distribution for the inputs, and shows that something similar to a Wiener-Hermite PC expansion can be derived as a reduced-rank GP. Whether it would be possible to generalise the result for d inputs remains a subject for future research. Relating reduced-rank GPs to all possible PC expansions in the generalised PC framework (for example the Legendre-chaos expansion in terms of uniformly distributed inputs) would also be a huge step in establishing theoretical links between the two approaches, if it is possible. This would undoubtedly be related to spectral decompositions of different covariance functions, or alternatively constructing the covariance function based on different polynomial families, and the suggested method of Särkkä et al. (2016) looks useful for this.

B.3. Chapter 3

Theoretical points

What theoretical results are available for the non-intrusive spectral projection (NISP) and regression methods for estimating the coefficients of the PC expansion, in the references that are cited? Do these reveal a theoretical preference for one method over the other?

On page 69 the NISP method for PC appears to interpolate the data. Can it be shown that this must be the case?

Response

There are little if no theoretical results available to compare the NISP and regression methods for estimating the PC coefficients. More frequent in the literature are examples where authors have used both techniques for benchmark simulators and have provided some empirical results as to which should be preferred. An example is Eldred et al. (2008), who conclude that both NISP and regression should be preferred over sampling. They suggest that NISP is best for low dimensional problems, regression is preferred for moderate dimensional problems, and for higher dimensional problems a sparse technique based on quadrature should be employed. There is definitely scope for a theoretical comparison of different PC methods, let alone a comparison with GP methods.

As to the second point, the NISP method interpolates the simulator design points since the design size n is chosen to be equal to the number of coefficients in the PC expansion N, and hence is a uniquely determined problem.

On quadrature for the non-intrusive spectral projection method for polynomial chaos

There is an assumption made in the thesis that NISP must be used in conjuction with (a) tensor production truncation of the basis expansion and (b) a tensor product of Gaussian quadrature points for $\{\mathbf{x}_i\}_{i=1}^n$. It is not clear that either of these restrictions are essential. As a result, it somewhat confounds the empirical results that compare regression PC to NISP PC, although it is accepted that this is not the focus of the thesis. For instance, what if the best PC method is tensor product truncation of the PC expansion, solved using NISP with Sobol points? This could be implemented, and in Chapter 5 it is shown how NISP need not be based on Gaussian quadrature

points. Indeed, this would be an interesting combination since it is known that digital nets (of which Sobol is one example) are a natural choice for integration of functions with dominating mixed smoothness, described by total order truncation (Dick and Pillichshammer, 2010).

Response

In this thesis, two methods for estimating the PC coefficients were considered. Regression, which was combined with a total order truncation of the PC expansion, and NISP (using tensor product quadrature), which was combined with a tensor product truncation of the PC expansion. These methods, and their associated truncation schemes, were chosen since they are popular approaches in the PC literature, and focused the scope for a comparison of PC with GP emulation. Whilst all combinations of the estimation method, truncation scheme, and quadrature points should have been considered for a complete analysis, many of them have not been considered in the PC literature and were not used in this work as a consequence. However, the combination of tensor product truncation, NISP, and Sobol points does seem like a interesting approach and is a potential avenue for future research.

Related (uncited) work

On page 91 it is stated that "PC surrogates do not provide any uncertainty information". However, Arnst et al. (2010) treats the PC coefficients as unknown random variables and takes a Bayesian approach to their estimation.

Response

The reference given was unfortunately overlooked in the literature review of PC methods given in Chapter 2. It seems like a valuable approach for estimating the coefficients in the presence of uncertainty, resulting in posterior distributions for the coefficients, and ultimately a PC surrogate which would provide uncertainty information. It would be beneficial in future work to compare this approach to the proposed probabilistic polynomial chaos method in Chapter 5, which also gives a posterior distribution for the coefficients.

B.4. Chapter 4

Performance assessment for the non-intrusive spectral projection method for polynomial chaos

On page 96 it is claimed that leave-one-out (LOO) cross-validation cannot be implemented for the NISP method because "all quadrature points must be used for the method to work". This is not awfully convincing; in Chapter 5, for instance, it is shown that other quadrature rules can be used within NISP.

Response

It is certainly the case that other quadrature rules can be used within NISP, and in Chapter 5 it was shown that Bayesian Monte Carlo and Bayesian Quasi-Monte Carlo quadrature rules are compatible. However, quadrature rules in their basic form must use all quadrature design points and weights for the estimation to work, so LOO cross-validation is not directly applicable. Ko and Wynn (2016) have proposed LOO cross-validation approaches for algebraic quadrature rules, with particular application to PC surrogate models, although this was considered an advanced approach and was not used in this thesis.

Quasi-Monte Carlo point sets versus Quasi-Monte Carlo point sequences

Quasi-Monte Carlo (QMC) point sequences (including the Sobol sequence) are by definition extensible, in contrast to QMC point sets (the latter must be re-constructed at each desired sample size, since a point set of size n need not contain a point set of size $m \ll n$). As a consequence of being extensible, QMC point sequences do not provide accurate integral estimates for all values of n. In particular, the Sobol sequence is only useful when $n = 2^m$ for some $m \in \mathbb{N}$. A background is given in Section 2 of Owen (2016).

Thus the values of n used in this thesis (which are not of the form 2^m) might not be suitable in conjunction with Sobol points. Intuitively, for $n \neq 2^m$ the Sobol points are not "evenly distributed" in $\mathcal{X}$.

Response

Sobol sequence designs were chosen because of their extensible property; it was possible to run the simulators at one large Sobol sequence design, and then use its subsets to train surrogate models on smaller designs whilst retaining the Sobol sequence property. This was done to save computational time and effort, rather than using, for example, several Latin Hypercube designs of different size. In the experiments in Chapter 4, the design sizes n were chosen to comply with the number of coefficients in PC expansions of different order. This naturally led to design sizes $n \neq 2^m$. Unfortunately, the above property of QMC point sequences was not known when carrying out the experiments, so it may be the case that some quadrature estimates are not accurate. However, the most important part of the experimental design problem was that PC and GP surrogate methods were built and compared on exactly the same designs in all cases. This was indeed the case, so the comparisons were considered fair.

Oakley and O'Hagan (2002) method

The algorithm presented on page 101 does not make sense — it is not possible just to sample from the posterior over $\eta(\mathcal{D}')$ given $\mathcal{D}$ and $\mathcal{Y}$?

Response

It would indeed be possible to sample from the posterior over $\eta(\mathcal{D}')$, although the size of the validation design, m, may result in computational problems when obtaining samples from the multivariate Student's-t distribution. In particular, numerical instabilities may be encountered when inverting the $m \times m$ covariance matrix. Instead, the method of Oakley and O'Hagan (2002) approximates samples from the posterior as a series of posterior mean functions which are quick to evaluate. The approximation is minimal with a suitable simulation design and provides a general estimator for any summary statistic of the simulator output. In the experiments in Chapter 4, $m = 1000$ so simulating from the posterior would be possible, but the approach of Oakley and O'Hagan (2002) was favoured as a rigorous approach for all validation metrics. It is expected that either approach would give similar results.

Confounding in the conclusion

On pages 112–113 it is stated that: "Finally, it is observed that the performance gap between PC and GP methods built on large tensor grid designs can largely be

eliminated with a switch to a Sobol sequence design of the same size". However, there is also a switch from regression to NISP when Sobol points are considered, so we do not know whether it is (*a*) the choice of points, (*b*) the fitting method (regression versus NISP), or (*c*) both that are most important.

Similar conclusions, such as: "Furthermore, the design type had considerable impact on which method was favoured", should be tempered with the fact that the performance could also be due to switching between regression and NISP.

Response

It is certainly the case that it may be the change in design type, the change in the fitting method for PC, or a combination of both that may be contributing to the results observed in this chapter. All conclusions such as those given above should be tempered with this point. Furthermore, the truncation scheme used for the PC expansion may have an effect because a tensor product truncation scheme uses more polynomial terms than a total order truncation scheme. Since there is little theoretical research to draw upon concerning the difference between regression and NISP (as stated earlier) however, it is difficult to discern whether it is the fitting method or the design type which is contributing most. Given that much of the surrogate modelling literature suggests that the size and type of design have large effects on the resulting surrogate (Loeppky et al., 2009; Chen et al., 2016), it is conjectured that the design points may be the dominating factor.

B.5. Chapter 5

Points for Bayesian quadrature

On pages 135–136 it is stated that: "As is the case for traditional numerical integration, the convergence rate of Bayesian quadrature for QMC samples is faster than that of the MC samples." This is not actually a correct statement. First, one should not directly compare a randomised to a non-randomised algorithm — different optimal rates can be achieved. Second, if the two are naïvely compared, the rate for Bayesian Monte Carlo has not yet proven to be faster than QMC (though it is believed to be faster).

Response

It is agreed that the above statement is not correct and perhaps the point needs to be clarified. It was intended to point out that for the example presented, the Bayesian quadrature (BQ) technique performed better when built using QMC samples rather than MC samples. This was simply measured visually — the BQ estimates were more accurate in terms of their mean (and with smaller variance) for the QMC samples. However, of course this is just for one realisation of the sampling strategy; to make a more rigorous statement about the convergence rates (even empirically) the experiment would need to be repeated. The point was alluding to the fact that for BQ, perhaps QMC samples would lead to faster convergence than when using MC samples, as known for standard quadrature. It was not known that this has not yet been proved.

Philosophical objections to probabilistic polynomial chaos

The examiners suggested that the probabilistic polynomial chaos (PPC) method being proposed did not make sense on the following philosophical (statistical) grounds:

1. The approach starts with a PC expansion in Equation (5.33). From the outset it can thus be argued that the uncertainty quantification being afforded to the surrogate is inappropriate, since in applications the simulator will not be of the polynomial form being posited in Equation (5.33). One work-around (which is discussed in a different context later, on page 148, and appears to have been considered in Schöbi et al. (2015)) would be to add a GP model discrepancy term to the end of Equation (5.33), but this is not pursued.

2. In order to estimate the PC coefficients, a second model is posited for the simulator in Equation (5.36) which is a GP. At this point two contrasting sets of statistical assumptions have been made about the simulator.

3. If we believe that the GP model assumptions in Equation (5.36) hold, why can the GP itself not be used as a surrogate for the simulator? It seems nonsensical that the fitted GP model should then be further approximated with a PC expansion — this can only introduce an additional source of approximation error.

Response

Concerning the first point, this seems a criticism of polynomial chaos on the whole, rather than of the proposed PPC methodology. Practitioners who use polynomial

chaos for uncertainty quantification in computer experiments routinely approximate the simulator using the polynomial form given in Equation (5.33). This strategy has been very successful in a wide range of cases, even when the simulator itself is not a polynomial. The polynomial order can be increased until the practitioner is satisfied with the approximation, or there are more advanced approaches one can use in more complex examples, as discussed in Chapter 2. As mentioned above and by Schöbi et al. (2015), to account for model discrepancy a GP model can be built on top of the PC approximation, although this is seldom considered in traditional PC so was not carried out here.

The purpose and philosophical reasoning for the proposed PPC methodology, which perhaps was not explored fully in Chapter 5, is to provide a probabilistic framework for traditional PC. That is, given that we want to approximate the simulator with a PC expansion in the form of Equation (5.33), can we add value to the current framework by deriving posterior distributions for coefficients of the expansion given runs of the simulator? The discrepancy between the PC approximation and the simulator unfortunately could not be considered within the scope of the thesis and is a direction for future research. The proposed way to answer the question above was to use Bayesian quadrature in conjunction with the NISP method for estimating the coefficients. This method assumes a GP model for the simulator to facilitate the derivation of the coefficient posteriors through BQ. Relating to the second point, it is correct that two contrasting sets of assumptions have been made about the simulator. Furthermore, as the third point suggests, at this point the GP model could instead be used as a surrogate for the simulator. However, this is not the purpose of the approach. The GP model is seen here as a tool to cast traditional PC in a probabilistic framework, and the proposed PPC methodology successfully does this. While it is admitted that there are some flaws with the approach, it is in its early developmental stage and there are several directions for future improvements.

Computational cost of polynomial chaos and Gaussian process emulation

It is claimed that GP is a more costly method than PC, due to the lack of need to estimate kernel hyperparameters in PC. The toy theoretical analysis given earlier in this appendix shows that this is not really the case — the Hermite polynomials in PC can be rescaled with hyperparameters μ, σ; i.e. $H_j(x) \to H_j\left(\frac{x-\mu}{\sigma}\right)$. The μ, σ can also be considered unknown and estimated (though not sure if this is widely acknowledged in the PC literature). Once this additional estimation task is taken into account, does PC have a computational advantage over GP? It is conceivable that it does still have an advantage over GP, since the model can be fitted in $\mathcal{O}(m^3)$

operations instead of $\mathcal{O}(n^3)$ due to the reduced-rank nature of PC.

Response

The toy theoretical example shows that the Hermite polynomials can be rescaled, leading to an additional estimation task in PC which was not considered in this work. Although this would increase the computational cost of PC to be on a similar level to that of GP emulation, is it something that is not done within the PC literature. Furthermore, as mentioned above the computational cost of PC would still better comparatively better due to its reduced-rank nature.

This is also related to what is known as the germ variable in the PC literature. The PC expansion in its traditional form uses well-known polynomial families evaluated at variables which have standard probability distributions (for example the standard Gaussian distribution). These variables are known as germs. If the inputs to the simulator cannot be modelled using standard probability distributions, an additional transformation to a germ variable must be made before fitting the PC expansion (although in practice this transformation is known and exact). This transformation could potentially involve an additional estimation procedure and could increase the computational cost of PC, as alluded to above.

B.6. Chapter 6

Related work

Related work is found in Roy et al. (2017), which appeared after the thesis was submitted.

Response

Roy et al. (2017) compare the approximation accuracy of PC and GP emulation, for a model of water flow with two inputs. Specifically, they use a PC expansion with the NISP quadrature approach and compare this to a GP emulator of principal components of the simulator output. Each surrogate is used to estimate statistical moments and the probability density function of the simulator output, for a single design of size $n = 121$. The accuracy is measured by comparing the estimated quantities to those from a large Monte Carlo simulation. Similar to the work in Chapter 4 and that which was published in Owen et al. (2017), they found that neither surrogate method outperforms the other unanimously, but advantages can be gained

in some cases. In particular, PC was more accurate in estimating the Sobol' indices (with the advantage that these can be obtained directly from the expansion coefficients), whereas GP emulators were better at estimating the multimodal probability density function of the simulator output.

It's promising to see that more examples of collaboration between the PC and GP communities are appearing in the literature. One of the aims of this thesis was to spark more interaction between practitioners of the two methodologies, through investigating comparisons and combinations of the approaches, and it is hoped that much more work will be carried out in the future.

Bibliography

Alkhatib, A. and P. King, 2014: Robust quantification of parametric uncertainty for surfactant-polymer flooding. *Computational Geosciences*, **18**, 77–101.

Andrianakis, I. and P. G. Challenor, 2012: The effect of the nugget on Gaussian process emulators of computer models. *Computational Statistics & Data Analysis*, **56**, 4215–4228.

Arnst, M., R. Ghanem, and C. Soize, 2010: Identification of Bayesian posteriors for coefficients of chaos expansions. *Journal of Computational Physics*, **229**, 3134–3154.

Askey, R. and J. Wilson, 1985: Some basic hypergeometric polynomials that generalize Jacobi polynomials. *Memoirs of the American Mathematical Society*, **319**.

Ba, S. and V. R. Joseph, 2012: Composite Gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, **6**, 1838–1860.

Babuška, I., F. Nobile, and R. Tempone, 2007: A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, **45**, 1005–1034.

Babuška, I., R. Tempone, and G. E. Zouraris, 2004: Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, **42**, 800–825.

Bartels, S., P. Hennig, and T. Germany, 2016: Probabilistic approximate least-squares. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 676–684.

Bastos, L. S. and A. O'Hagan, 2009: Diagnostics for Gaussian process emulators. *Technometrics*, **51**, 425–438.

Bayarri, M. J., J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu, 2007: A framework for validation of computer models. *Technometrics*, **49**, 138–154.

Bellouin, N., W. Collins, I. Culverwell, P. Halloran, S. Hardiman, T. Hinton, C. Jones, R. McDonald, A. McLaren, F. O'Connor, et al., 2011: The HadGEM2 family of Met Office Unified Model climate configurations. *Geoscientific Model Development*, **4**, 723–757.

Berveiller, M., Y. Le Pape, B. Sudret, and F. Perrin, 2012: Updating the long-term strains in concrete containment vessels by using Markov chain Monte Carlo simulation and polynomial chaos expansions. *Structure and Infrastructure Engineering*, **8**, 425–440.

Berveiller, M., B. Sudret, and M. Lemaire, 2004: Presentation of two methods for computing the response coefficients in stochastic finite element analysis. *Proceedings of the 9th ASCE Specialty Conference on Probabilistic Mechanics and Structural Reliability, Albuquerque, USA*.

— 2006: Stochastic finite element: a non intrusive approach by regression. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, **15**, 81–92.

Best, M., M. Pryor, D. Clark, G. Rooney, R. Essery, C. Ménard, J. Edwards, M. Hendry, A. Porson, N. Gedney, et al., 2011: The joint UK land environment simulator (JULES), model description—part 1: energy and water fluxes. *Geoscientific Model Development*, **4**, 677–699.

Blatman, G. and B. Sudret, 2010a: An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, **25**, 183–197.

— 2010b: Efficient computation of global sensitivity indices using sparse polynomial chaos expansions. *Reliability Engineering & System Safety*, **95**, 1216–1229.

— 2011: Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, **230**, 2345–2367.

Blight, B. and L. Ott, 1975: A Bayesian approach to model inadequacy for polynomial regression. *Biometrika*, **62**, 79–88.

Bower, R., A. Benson, R. Malbon, J. Helly, C. Frenk, C. Baugh, S. Cole, and C. G. Lacey, 2006: Breaking the hierarchy of galaxy formation. *Monthly Notices of the Royal Astronomical Society*, **370**, 645–655.

Box, G. E. and D. R. Cox, 1964: An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 211–252.

Briol, F.-X., C. Oates, M. Girolami, and M. A. Osborne, 2015a: Frank-Wolfe Bayesian quadrature: Probabilistic integration with theoretical guarantees. *Advances in Neural Information Processing Systems*, 1162–1170.

Briol, F.-X., C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic, 2015b: Probabilistic integration: A role for statisticians in numerical analysis? *arXiv preprint arXiv:1512.00933*.

Brooks, S., A. Gelman, G. Jones, and X.-L. Meng, 2011: *Handbook of Markov Chain Monte Carlo*. CRC press.

Byrd, R. H., P. Lu, J. Nocedal, and C. Zhu, 1995: A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, **16**, 1190–1208.

Caflisch, R. E., 1998: Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, **7**, 1–49.

Cameron, R. H. and W. T. Martin, 1947: The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals. *Annals of Mathematics*, **48**, 385–392.

Chen, H., J. L. Loeppky, J. Sacks, W. J. Welch, et al., 2016: Analysis methods for computer experiments: How to assess and what counts? *Statistical Science*, **31**, 40–60.

Chihara, T. S., 2011: *An Introduction to Orthogonal Polynomials*. Courier Corporation.

Chkrebtii, O. A., D. A. Campbell, B. Calderhead, M. A. Girolami, et al., 2016: Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, **11**, 1239–1267.

Choi, S.-K., R. V. Grandhi, R. A. Canfield, and C. L. Pettit, 2004: Polynomial chaos expansion with Latin Hypercube sampling for estimating response variability. *AIAA Journal*, **42**, 1191–1198.

Chorin, A. J., 1971: Hermite expansions in Monte-Carlo computation. *Journal of Computational Physics*, **8**, 472–482.

— 1974: Gaussian fields and random flow. *Journal of Fluid Mechanics*, **63**, 21–32.

Clark, D., L. Mercado, S. Sitch, C. Jones, N. Gedney, M. Best, M. Pryor, G. Rooney, R. Essery, E. Blyth, et al., 2011: The joint UK land environment simulator (JULES), model description—part 2: carbon fluxes and vegetation dynamics. *Geoscientific Model Development*, **4**, 701–722.

Cockayne, J., C. Oates, T. Sullivan, and M. Girolami, 2017: Bayesian probabilistic numerical methods. *arXiv preprint arXiv:1702.03673*.

Conrad, P. R., M. Girolami, S. Särkkä, A. Stuart, and K. Zygalakis, 2015: Probability measures for numerical solutions of differential equations. *arXiv preprint arXiv:1506.04592*.

Conrad, P. R. and Y. M. Marzouk, 2013: Adaptive Smolyak pseudospectral approximations. *SIAM Journal on Scientific Computing*, **35**, A2643–A2670.

Conti, S. and A. O'Hagan, 2010: Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, **140**, 640–651.

Craig, P., M. Goldstein, A. Seheult, and J. Smith, 1996: Bayes linear strategies for matching hydrocarbon reservoir history. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 5, 69–95.

Craig, P. S., M. Goldstein, J. C. Rougier, and A. H. Seheult, 2001: Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, **96**, 717–729.

Crow, S. and G. Canavan, 1970: Relationship between a Wiener-Hermite expansion and an energy cascade. *Journal of Fluid Mechanics*, **41**, 387–403.

Currin, C., T. Mitchell, M. Morris, and D. Ylvisaker, 1991: Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, **86**, 953–963.

Debusschere, B. J., H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître, 2004: Numerical challenges in the use of polynomial chaos representations for stochastic processes. *SIAM Journal on Scientific Computing*, **26**, 698–719.

Der Kiureghian, A. and P.-L. Liu, 1986: Structural reliability under incomplete probability information. *Journal of Engineering Mechanics*, **112**, 85–104.

Diaconis, P., 1988: Bayesian numerical analysis. *Statistical Decision Theory and Related Topics IV*, Springer-Verlag, New York, volume 1, 163–175.

DiazDelaO, F. and S. Adhikari, 2009: Coupling polynomial chaos expansions with Gaussian process emulators: an introduction. *Proceedings of the IMAC-XXVII, Orlando, Florida, USA*.

— 2011: Gaussian process emulators for the stochastic finite element method. *International Journal for Numerical Methods in Engineering*, **87**, 521–540.

Dick, J. and F. Pillichshammer, 2010: *Digital nets and sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press.

Doostan, A. and H. Owhadi, 2011: A non-adapted sparse approximation of PDEs with stochastic inputs. *Journal of Computational Physics*, **230**, 3015–3034.

Efron, B., T. Hastie, I. Johnstone, R. Tibshirani, et al., 2004: Least angle regression. *The Annals of Statistics*, **32**, 407–499.

Efron, B. and R. J. Tibshirani, 1994: *An Introduction to the Bootstrap*. CRC Press.

Eldred, M. S. and J. Burkardt, 2009: Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification. *Proceedings of the 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*.

Eldred, M. S., C. G. Webster, and P. Constantine, 2008: Evaluation of non-intrusive approaches for Wiener-Askey generalized polynomial chaos. *Proceedings of the 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.

Erdös, P. and M. Kac, 1940: The Gaussian law of errors in the theory of additive number theoretic functions. *American Journal of Mathematics*, **62**, 738–742.

Faraway, J. J., 2006: *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*, volume 124. CRC press.

Fichtl, E. D. and A. K. Prinja, 2011: The stochastic collocation method for radiation transport in random media. *Journal of Quantitative Spectroscopy and Radiative Transfer*, **112**, 646–659.

Forrester, A. I., A. Sóbester, and A. J. Keane, 2007: Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, volume 463, 3251–3269.

Fricker, T. E., J. E. Oakley, and N. M. Urban, 2013: Multivariate Gaussian process emulators with nonseparable covariance structures. *Technometrics*, **55**, 47–56.

Ghanem, R., 1998: Probabilistic characterization of transport in heterogeneous media. *Computer Methods in Applied Mechanics and Engineering*, **158**, 199–220.

Ghanem, R. and S. Dham, 1998: Stochastic finite element analysis for multiphase flow in heterogeneous porous media. *Transport in Porous Media*, **32**, 239–262.

Ghanem, R., J. Red-Horse, and A. Sarkar, 2000: Modal properties of a space-frame with localized system uncertainties. *Proceedings of the 8th ASCE Specialty Conference of Probabilistic Mechanics and Structural Reliability*, Citeseer.

Ghanem, R. and P. Spanos, 1990: Polynomial chaos in stochastic finite elements. *Journal of Applied Mechanics*, **57**, 197–202.

Ghanem, R. G. and P. D. Spanos, 1991a: Spectral stochastic finite-element formulation for reliability analysis. *Journal of Engineering Mechanics*, **117**, 2351–2372.

— 1991b: *Stochastic Finite Elements: A Spectral Approach*. Springer New York.

Ghiocel, D. M. and R. G. Ghanem, 2002: Stochastic finite-element analysis of seismic soil-structure interaction. *Journal of Engineering Mechanics*, **128**, 66–77.

Gilli, L., D. Lathouwers, J. L. Kloosterman, T. van der Hagen, A. Koning, and D. Rochman, 2013: Uncertainty quantification for criticality problems using non-intrusive and adaptive polynomial chaos techniques. *Annals of Nuclear Energy*, **56**, 71–80.

Goldstein, M. and J. Rougier, 2006: Bayes linear calibrated prediction for complex systems. *Journal of the American Statistical Association*, **101**, 1132–1143.

Gordon, C., C. Cooper, C. A. Senior, H. Banks, J. M. Gregory, T. C. Johns, J. F. Mitchell, and R. A. Wood, 2000: The simulation of SST, sea ice extents and ocean heat transports in a version of the Hadley Centre coupled model without flux adjustments. *Climate Dynamics*, **16**, 147–168.

Gramacy, R. B. and D. W. Apley, 2015: Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, **24**, 561–578.

Gramacy, R. B. and H. K. Lee, 2008: Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, **103**, 1119–1130.

Gu, M. and J. O. Berger, 2016: Parallel partial Gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, **10**, 1317–1347.

Guillas, S., J. Rougier, A. Maute, A. Richmond, and C. Linkletter, 2009: Bayesian calibration of the thermosphere-ionosphere electrodynamics general circulation model (TIE-GCM). *Geoscientific Model Development*, **2**, 137–144.

Gunter, T., M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts, 2014: Sampling for inference in probabilistic models with fast Bayesian quadrature. *Advances in Neural Information Processing Systems*, 2789–2797.

Hagan, M. T., H. B. Demuth, M. H. Beale, and O. De Jesús, 1996: *Neural Network Design*, volume 20. PWS publishing company, Boston.

Haylock, R. and A. O'Hagan, 1996: On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 5, 629–637.

Hennig, P., 2015: Probabilistic interpretation of linear solvers. *SIAM Journal on Optimization*, **25**, 234–260.

Hennig, P. and M. Kiefel, 2013: Quasi-Newton methods: A new direction. *Journal of Machine Learning Research*, **14**, 843–865.

Hennig, P., M. A. Osborne, and M. Girolami, 2015: Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, volume 471, 20150142.

Higdon, D., J. Gattiker, B. Williams, and M. Rightley, 2008: Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, **103**, 570–583.

Higdon, D., M. Kennedy, J. C. Cavendish, J. A. Cafeo, and R. D. Ryne, 2004: Combining field data and computer simulations for for calibration and prediction. *SIAM Journal on Scientific Computing*, **26**, 448–466.

Higdon, D., J. Swall, and J. Kern, 1998: Non-stationary spatial modeling. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 5, 761–768.

Hitzl, D. L. and F. H. Maltz, 1980: Adaptive estimation procedures for multiparameter Monte Carlo computations. *Journal of Computational Physics*, **37**, 218–241.

Hosder, S., R. W. Walters, and M. Balch, 2007: Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables. *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.

Hull, T. and J. R. Swenson, 1966: Tests of probabilistic models for propagation of roundoff errors. *Communications of the ACM*, **9**, 108–113.

Isukapalli, S., A. Roy, and P. Georgopoulos, 1998: Stochastic response surface methods (SRSMs) for uncertainty propagation: application to environmental and biological systems. *Risk Analysis*, **18**, 351–363.

Jakeman, J. D., M. S. Eldred, and K. Sargsyan, 2015: Enhancing 1-minimization estimates of polynomial chaos expansions using basis selection. *Journal of Computational Physics*, **289**, 18–34.

Jakeman, J. D., A. Narayan, and D. Xiu, 2013: Minimal multi-element stochastic collocation for uncertainty quantification of discontinuous functions. *Journal of Computational Physics*, **242**, 790–808.

Keese, A. and H. G. Matthies, 2003: Sparse quadrature as an alternative to Monte Carlo for stochastic finite element techniques. *Proceedings in Applied Mathematics and Mechanics*, **3**, 493–494.

Kennedy, M., 1998: Bayesian quadrature with non-normal approximating functions. *Statistics and Computing*, **8**, 365–375.

Kennedy, M. C. and A. O'Hagan, 2000: Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, **87**, 1–13.

— 2001: Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 425–464.

Kersaudy, P., B. Sudret, N. Varsier, O. Picon, and J. Wiart, 2015: A new surrogate modeling technique combining Kriging and polynomial chaos expansions — application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, **286**, 103–117.

Kimeldorf, G. S. and G. Wahba, 1970: A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, **41**, 495–502.

Kirkpatrick, S. W., 2000: Development and validation of high fidelity vehicle crash simulation models. Technical report, SAE Technical Paper.

Knio, O. and O. Le Maître, 2006: Uncertainty propagation in CFD using polynomial chaos decomposition. *Fluid Dynamics Research*, **38**, 616–640.

Ko, J. and H. P. Wynn, 2016: The algebraic method in quadrature for uncertainty quantification. *SIAM/ASA Journal on Uncertainty Quantification*, **4**, 331–357.

Kong, A., P. McCullagh, X.-L. Meng, D. Nicolae, and Z. Tan, 2003: A theory of statistical models for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**, 585–604.

Krige, D., 1951: A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of Chemical, Metallurgical, and Mining Society of South Africa*, **52**, 119–139.

Kynn, M., 2008: The 'heuristics and biases' bias in expert elicitation. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **171**, 239–264.

Le Gratiet, L. and C. Cannamela, 2015: Cokriging-based sequential design strategies using fast cross-validation techniques for multi-fidelity computer codes. *Technometrics*, **57**, 418–427.

Le Maître, O., O. Knio, H. Najm, and R. Ghanem, 2004a: Uncertainty propagation using Wiener-Haar expansions. *Journal of Computational Physics*, **197**, 28–57.

Le Maître, O., H. Najm, R. Ghanem, and O. Knio, 2004b: Multi-resolution analysis of Wiener-type uncertainty propagation schemes. *Journal of Computational Physics*, **197**, 502–531.

Le Maître, O. P., O. M. Knio, H. N. Najm, and R. G. Ghanem, 2001: A stochastic projection method for fluid flow: 1. Basic formulation. *Journal of Computational Physics*, **173**, 481–511.

Le Maître, O. P., M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, 2002: A stochastic projection method for fluid flow: 2. Random process. *Journal of Computational Physics*, **181**, 9–44.

Liu, D., A. Litvinenko, C. Schillings, and V. Schulz, 2017: Quantification of airfoil geometry-induced aerodynamic uncertainties—comparison of approaches. *SIAM/ASA Journal on Uncertainty Quantification*, **5**, 334–352.

Loeppky, J. L., J. Sacks, and W. J. Welch, 2009: Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, **51**, 366–376.

Loève, M., 1977: *Probability Theory*. Springer-Verlag, New York, 4 edition.

Mahsereci, M. and P. Hennig, 2015: Probabilistic line searches for stochastic optimization. *Advances in Neural Information Processing Systems*, 181–189.

Maltz, F. and D. Hitzl, 1979: Variance reduction in Monte Carlo computations using multi-dimensional Hermite polynomials. *Journal of Computational Physics*, **32**, 345–376.

Marzouk, Y. M. and H. N. Najm, 2009: Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems. *Journal of Computational Physics*, **228**, 1862–1902.

Marzouk, Y. M., H. N. Najm, and L. A. Rahn, 2007: Stochastic spectral methods for efficient Bayesian solution of inverse problems. *Journal of Computational Physics*, **224**, 560–586.

Matheron, G., 1963: Principles of geostatistics. *Economic Geology*, **58**, 1246–1266.

Mattern, J. P., K. Fennel, and M. Dowd, 2012: Estimating time-dependent parameters for a biological ocean model using an emulator approach. *Journal of Marine Systems*, **96**, 32–47.

McFarland, J., S. Mahadevan, L. Swiler, and A. Giunta, 2007: Bayesian calibration of the QASPR simulation. *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.

McKay, M. D., R. J. Beckman, and W. J. Conover, 1979: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**, 239–245.

Meecham, W. C. and D.-T. Jeng, 1968: Use of the Wiener-Hermite expansion for nearly normal turbulence. *Journal of Fluid Mechanics*, **32**, 225–249.

Minka, T., 2000: Deriving quadrature rules from Gaussian processes. Technical report, Statistics Department, Carnegie Mellon University.

Mortensen, R. E. and K. P. Haggerty, 1988: A stochastic computer model for heating and cooling loads. *IEEE Transactions on Power Systems*, **3**, 1213–1219.

Narayan, A., C. Gittelson, and D. Xiu, 2014: A stochastic collocation algorithm with multifidelity models. *SIAM Journal on Scientific Computing*, **36**, 495–521.

Neal, R. M., 1998: Regression and classification using Gaussian process priors. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 6, 475–502.

Niederreiter, H., 1988: Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*, **30**, 51–70.

Nobile, F., R. Tempone, and C. G. Webster, 2008a: A anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, **46**, 2411–2442.

— 2008b: A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, **46**, 2309–2345.

Oakley, J., 2011: Modelling with deterministic computer models. *Simplicity, Complexity and Modelling*, M. Christie, A. Cliffe, P. Dawid, and S. S. Senn, eds., John Wiley & Sons, Ltd.

Oakley, J. and A. O'Hagan, 2002: Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, **89**, 769–784.

Oakley, J. E. and A. O'Hagan, 2004: Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **66**, 751–769.

Oates, C., S. Niederer, A. Lee, F.-X. Briol, and M. Girolami, 2016: Probabilistic models for integration error in the assessment of functional cardiac models. *arXiv preprint arXiv:1606.06841*.

O'Hagan, A., 1978: Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society: Series B (Methodological)*, **40**, 1–42.

— 1991: Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, **29**, 245–260.

— 1992: Some Bayesian numerical analysis. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 4, 345–363.

— 1998: Uncertainty analysis and other inference tools for complex computer codes. *Bayesian Statistics*, J. Bernardo, J. Berger, A. Dawid, and A. Smith, eds., Oxford University Press, volume 6.

— 2006: Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, **91**, 1290–1300.

— 2013: Polynomial chaos: A tutorial and critique from a statistician's perspective, submitted to SIAM/ASA Journal on Uncertainty Quantification.

Oladyshkin, S. and W. Nowak, 2012: Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering & System Safety*, **106**, 179–190.

Osborne, M., R. Garnett, Z. Ghahramani, D. K. Duvenaud, S. J. Roberts, and C. E. Rasmussen, 2012a: Active learning of model evidence using Bayesian quadrature. *Advances in Neural Information Processing Systems*, 46–54.

Osborne, M. A., R. Garnett, S. J. Roberts, C. Hart, S. Aigrain, N. Gibson, and S. Aigrain, 2012b: Bayesian quadrature for ratios. *AISTATS*, 832–840.

Oughton, R. H. and P. S. Craig, 2016: Hierarchical emulation: A method for modeling and comparing nested simulators. *SIAM/ASA Journal on Uncertainty Quantification*, **4**, 495–519.

Owen, A. B., 2016: A constraint on extensible quadrature rules. *Numerische Mathematik*, **132**, 511–518.

Owen, N. E., P. Challenor, P. P. Menon, and S. Bennani, 2017: Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators. *SIAM/ASA Journal on Uncertainty Quantification*, **5**, 403–435.

Owhadi, H., 2015: Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, **13**, 812–828.

Park, J.-S. and J. Baek, 2001: Efficient computation of maximum likelihood estimators in a spatial linear model with power exponential covariogram. *Computers & Geosciences*, **27**, 1–7.

Patel, J. K. and C. B. Read, 1996: *Handbook of the Normal Distribution*, volume 150. CRC Press.

Picheny, V. and D. Ginsbourger, 2013: A nonstationary space-time Gaussian process model for partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification*, **1**, 57–78.

Poincaré, H., 1912: *Calcul des Probabilités*. Gauthier-Villars.

Qian, P. Z. and C. J. Wu, 2008: Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics*, **50**, 192–204.

R Core Team, 2016: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
URL `http://www.R-project.org/`

Raoult, N. M., T. E. Jupp, P. M. Cox, and C. M. Luke, 2016: Land-surface parameter optimisation using data assimilation techniques: the adJULES system V1.0. *Geoscientific Model Development*, **9**, 2833–2852.

Rasmussen, C. E. and Z. Ghahramani, 2002: Bayesian Monte Carlo. *Advances in Neural Information Processing Systems*, 489–496.

Rasmussen, C. E. and C. K. Williams, 2006: *Gaussian Processes for Machine Learning*. MIT Press.

Ray, J., Z. Hou, M. Huang, K. Sargsyan, and L. Swiler, 2015: Bayesian calibration of the community land model using surrogates. *SIAM/ASA Journal on Uncertainty Quantification*, **3**, 199–233.

Reagan, M. T., H. N. Najm, R. G. Ghanem, and O. M. Knio, 2003: Uncertainty quantification in reacting-flow simulations through non-intrusive spectral projection. *Combustion and Flame*, **132**, 545–555.

Reich, B. J., C. B. Storlie, and H. D. Bondell, 2012: Variable selection in Bayesian smoothing spline ANOVA models: Application to deterministic computer codes. *Technometrics*, **51**, 110–120.

Rosenblatt, M., 1952: Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, **23**, 470–472.

Rougier, J., 2008: Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, **17**, 827–843.

Rougier, J., S. Guillas, A. Maute, and A. D. Richmond, 2009a: Expert knowledge and multivariate emulation: The thermosphere-ionosphere electrodynamics general circulation model (TIE-GCM). *Technometrics*, **51**, 414–424.

Rougier, J., D. M. Sexton, J. M. Murphy, and D. Stainforth, 2009b: Analyzing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments. *Journal of Climate*, **22**, 3540–3557.

Roustant, O., D. Ginsbourger, and Y. Deville, 2012: DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. *Journal of Statistical Software*, **51**, 1–55.
URL `http://www.jstatsoft.org/v51/i01/`

Roy, P. T., N. E. Moçayd, S. Ricci, J.-C. Jouhaud, N. Goutal, M. De Loco, and M. C. Rochoux, 2017: Comparison of polynomial chaos and Gaussian process surrogates for uncertainty quantification and correlation estimation of spatially distributed open-channel steady flows. *arXiv preprint arXiv:1705.01440*.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn, 1989: Design and analysis of computer experiments. *Statistical Science*, **4**, 409–435.

Saltelli, A., K. Chan, and E. M. Scott, 2009: *Sensitivity Analysis*. Wiley New York.

Sammut, C. and G. I. Webb, 2011: *Encyclopedia of Machine Learning*. Springer Science & Business Media.

Sampson, P. D. and P. Guttorp, 1992: Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, **87**, 108–119.

Sansó, B. and C. Forest, 2009: Statistical calibration of climate system properties. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **58**, 485–503.

Santner, T. J., B. J. Williams, and W. I. Notz, 2003: *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York.

Sargsyan, K., C. Safta, H. N. Najm, B. J. Debusschere, D. Ricciuto, and P. Thornton, 2014: Dimensionality reduction for complex models via Bayesian compressive sensing. *International Journal for Uncertainty Quantification*, **4**, 63–93.

Särkkä, S., J. Hartikainen, L. Svensson, and F. Sandblom, 2016: On the relation between Gaussian process quadratures and sigma-point methods. *Journal of Advances in Information Fusion*, **11**, 31–46.

Schmidt, A. M. and A. O'Hagan, 2003: Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**, 743–758.

Schober, M., D. K. Duvenaud, and P. Hennig, 2014: Probabilistic ODE solvers with Runge-Kutta means. *Advances in Neural Information Processing Systems*, 739–747.

Schöbi, R. and B. Sudret, 2015: Imprecise structural reliability analysis using PC-Kriging. *Proceedings of the 25th European Safety and Reliability Conference (ESREL2015)*.

Schöbi, R., B. Sudret, and S. Marelli, 2016: Rare event estimation using polynomial-chaos kriging. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, D4016002.

Schöbi, R., B. Sudret, and J. Wiart, 2015: Polynomial-chaos-based Kriging. *International Journal for Uncertainty Quantification*, **5**.

Schwab, C. and R. Todor, 2003: Sparse finite elements for elliptic problems with stochastic data. *Numerische Mathematik*, **95**, 707–734.

Shi, T., M. Belkin, and B. Yu, 2009: Data spectroscopy: Eigenspaces of convolution operators and clustering. *The Annals of Statistics*, 3960–3984.

Silverman, B. W., 1986: *Density Estimation for Statistics and Data Analysis*, volume 26. CRC press.

Skilling, J., 1992: Bayesian solution of ordinary differential equations. *Maximum entropy and Bayesian methods*, Springer, 23–37.

Smith, R. C., 2013: *Uncertainty Quantification: Theory, Implementation, and Applications*, volume 12. SIAM.

Smolyak, S. A., 1963: Quadrature and interpolation formulas for tensor products of certain classes of functions. *Doklady Akademiia Nauk SSSR*, volume 4, 240–243.

Soize, C., 2015: Polynomial chaos expansion of a multimodal random vector. *SIAM/ASA Journal on Uncertainty Quantification*, **3**, 34–60.

Soize, C. and R. Ghanem, 2004: Physical systems with random uncertainties: chaos representations with arbitrary probability measure. *SIAM Journal on Scientific Computing*, **26**, 395–410.

Sraj, I., S. E. Zedler, O. M. Knio, C. S. Jackson, and I. Hoteit, 2016: Polynomial chaos-based Bayesian inference of k-profile parameterization in a general circulation model of the tropical pacific. *Monthly Weather Review*, **144**, 4621–4640.

Stein, M. L., 1999: *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, New York.

Steinwart, I. and A. Christmann, 2008: *Support Vector Machines*. Springer Science & Business Media.

Stuart, A. M. and A. L. Teckentrup, 2016: Posterior consistency for Gaussian process approximations of Bayesian posterior distributions. *arXiv preprint arXiv:1603.02004*.

Sudret, B., 2008: Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, **98**, 964–979.

Sullivan, T. J., 2015: *Introduction to Uncertainty Quantification*, volume 63. Springer.

Sun, R., 2006: *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press.

Tagade, P. M. and H.-L. Choi, 2014: A generalized polynomial chaos-based method for efficient Bayesian calibration of uncertain computational models. *Inverse Problems in Science and Engineering*, **22**, 602–624.

Teckentrup, A. L., P. Jantsch, C. G. Webster, and M. Gunzburger, 2015: A multilevel stochastic collocation method for partial differential equations with random input data. *SIAM/ASA Journal on Uncertainty Quantification*, **3**, 1046–1074.

Urban, N. M. and T. E. Fricker, 2010: A comparison of Latin hypercube and grid ensemble designs for the multivariate emulation of an earth system model. *Computers & Geosciences*, **36**, 746–755.

Vernon, I., M. Goldstein, R. G. Bower, et al., 2010: Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, **5**, 619–669.

Wan, X. and G. E. Karniadakis, 2005: An adaptive multi-element generalized polynomial chaos method for stochastic differential equations. *Journal of Computational Physics*, **209**, 617–642.

Welch, W. J., R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, 1992: Screening, predicting, and computer experiments. *Technometrics*, **34**, 15–25.

Wiener, N., 1938: The homogeneous chaos. *American Journal of Mathematics*, **60**, 897–936.

Williams, B., D. Higdon, J. Gattiker, L. Moore, M. McKay, S. Keller-McNulty, et al., 2006: Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis*, **1**, 765–792.

Williamson, D. and A. T. Blaker, 2014: Evolving Bayesian emulators for structured chaotic time series, with application to large climate models. *SIAM/ASA Journal on Uncertainty Quantification*, **2**, 1–28.

Williamson, D., A. T. Blaker, C. Hampton, and J. Salter, 2015: Identifying and removing structural biases in climate models with history matching. *Climate Dynamics*, **45**, 1299–1324.

Williamson, D., M. Goldstein, L. Allison, A. Blaker, P. Challenor, L. Jackson, and K. Yamazaki, 2013: History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate Dynamics*, **41**, 1703–1729.

Williamson, D., M. Goldstein, and A. Blaker, 2012: Fast linked analyses for scenario-based hierarchies. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **61**, 665–691.

Winokur, J., P. Conrad, I. Sraj, O. Knio, A. Srinivasan, W. C. Thacker, Y. Marzouk, and M. Iskandarani, 2013: A priori testing of sparse adaptive polynomial chaos expansions using an ocean general circulation model database. *Computational Geosciences*, **17**, 899–911.

Winokur, J., D. Kim, F. Bisetti, O. Le Maître, and O. Knio, 2016: Sparse pseudo spectral projection methods with directional adaptation for uncertainty quantification. *Journal of Scientific Computing*, **68**, 596–623.

Xiu, D., 2009: Fast numerical methods for stochastic computations: A review. *Communications in Computational Physics*, **5**, 242–272.

— 2010: *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press.

Xiu, D. and J. S. Hesthaven, 2005: High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, **27**, 1118–1139.

Xiu, D. and G. E. Karniadakis, 2002: The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, **24**, 619–644.

— 2003: Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, **187**, 137–167.

Xiu, D. and S. J. Sherwin, 2007: Parametric uncertainty analysis of pulse wave propagation in a model of a human arterial network. *Journal of Computational Physics*, **226**, 1385–1407.